CFM 03110 US
Appln. No. 10/600,843
GAU: N4A

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００２年　６月２８日

出願番号
Application Number: 特願２００２－１９１１２９

[ST. 10/C]: [ＪＰ２００２－１９１１２９]

出願人
Applicant(s): キヤノン株式会社

２００３年　７月１０日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００３－３０５５５９８

【書類名】　特許願
【整理番号】　4741005
【提出日】　平成14年 6月28日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 9/00
【発明の名称】　ログ取得方法およびプログラム、記憶媒体
【請求項の数】　21
【発明者】
【住所又は居所】　東京都大田区下丸子３丁目３０番２号 キヤノン株式会社内
【氏名】　三原　誠
【特許出願人】
【識別番号】　000001007
【氏名又は名称】　キヤノン株式会社
【代理人】
【識別番号】　100076428
【弁理士】
【氏名又は名称】　大塚　康徳
【電話番号】　03-5276-3241
【選任した代理人】
【識別番号】　100112508
【弁理士】
【氏名又は名称】　高柳　司郎
【電話番号】　03-5276-3241
【選任した代理人】
【識別番号】　100115071
【弁理士】
【氏名又は名称】　大塚　康弘
【電話番号】　03-5276-3241

【選任した代理人】

【識別番号】　100116894

【弁理士】

【氏名又は名称】　木村　秀二

【電話番号】　03-5276-3241

【手数料の表示】

【予納台帳番号】　003458

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0102485

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　ログ取得方法およびプログラム、記憶媒体

【特許請求の範囲】

【請求項１】　所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記選択する工程において選択された関数を呼び出す際の所定の情報をログとして記録する工程と、

前記選択する工程において選択された関数の実行結果を受け取った際の所定の情報をログとして記録する工程と

を備えることを特徴とするログ取得方法。

【請求項２】　所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、該関数の実行結果を受け取った際の所定の情報とをログとして記録する工程と、

前記選択する工程において選択された関数について、ログの記録を停止する工程と

を備えることを特徴とするログ取得方法。

【請求項３】　前記ログ取得のための関数は、前記選択する工程において選択された関数の実行時のエラーの有無を判断する工程を更に備え、

前記ログとして記録する工程は、前記エラーの有無を判断する工程によりエラー有りと判断された場合、前記所定の情報をログとして記録することを特徴とする請求項１に記載のログ取得方法。

【請求項４】　前記ログ取得のための関数は、前記選択する工程において選択された関数の実行時のエラーの有無を判断する工程を更に備え、

前記ログの記録を停止する工程は、前記エラーの有無を判断する工程によりエラー有りと判断された場合、ログの記録を停止することを特徴とする請求項２に記載のログ取得方法。

【請求項５】　所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記選択する工程において選択されたメソッドを呼び出す際の所定の情報をログとして記録する工程と、

前記選択する工程において選択されたメソッドの実行結果を受け取った際の所定の情報をログとして記録する工程と

を備えることを特徴とするログ取得方法。

【請求項６】　所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、該関数の実行結果を受け取った際の所定の情報とをログとして記録する工程と、

前記選択する工程において選択されたメソッドについて、ログの記録を停止する工程と

を備えることを特徴とするログ取得方法。

【請求項７】　前記ログ取得のための関数は、前記選択する工程において選択されたメソッドの実行時のエラーの有無を判断する工程を更に備え、

前記ログとして記録する工程は、前記エラーの有無を判断する工程によりエラー有りと判断された場合、前記所定の情報をログとして記録することを特徴とする請求項５に記載のログ取得方法。

【請求項８】　前記ログ取得のための関数は、前記選択する工程において選択されたメソッドの実行時のエラーの有無を判断する工程を更に備え、

前記ログの記録を停止する工程は、前記エラーの有無を判断する工程によりエラー有りと判断された場合、ログの記録を停止することを特徴とする請求項６に記載のログ取得方法。

【請求項９】　ＣＯＭサーバがエクスポートするインターフェースと、該インターフェースに属するメソッドとをツリー構造で表示する工程を更に備え、

前記選択する工程は、前記表示を介して選択されることを特徴とする請求項５または６に記載のログ取得方法。

【請求項１０】　前記選択する工程は、メソッド単位で選択可能であることを特徴とする請求項９に記載のログ取得方法。

【請求項１１】　前記選択する工程は、前記インターフェースを選択することで、該インターフェースに属する全てのメソッドを選択可能であることを特徴とする請求項９に記載のログ取得方法。

【請求項１２】　所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数

のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、日付毎にログを記録可能であることを特徴とするログ取得方法。

【請求項１３】　所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、ログのサイズが一定サイズ以上になった場合に、新たに別ファイルを作成するとともに、作成日時の古いログを削除することを特徴とするログ取得方法。

【請求項１４】　所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った

実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、ログの個数が一定数以上になった場合に、新たに別ファイルを作成するとともに、作成日時の古いログを削除することを特徴とするログ取得方法。

【請求項１５】　所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとしてメモリに格納する工程と、を備え、

前記ログとしてメモリに格納する工程は、メモリ上のログの個数が一定数以上になった場合に、該メモリ上のログをディスク装置に移動させ、保存させることを特徴とするログ取得方法。

【請求項１６】　所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、日付毎にログを記録可能であることを特徴とするログ取得方法。

【請求項１７】　所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、ログのサイズが一定サイズ以上になった場合に、新たに別ファイルを作成するとともに、作成日時の古いログを削除することを特徴とするログ取得方法。

【請求項１８】　所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、ログの個数が一定数以上になった場合に、新たに別ファイルを作成するとともに、作成日時の古いログを削除することを特徴とするログ取得方法。

【請求項１９】　所定の処理を行うメソッドを備えるプログラムの実行中の

ログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとしてメモリに格納する工程と、を備え、

前記ログとしてメモリに格納する工程は、メモリ上のログの個数が一定数以上になった場合に、該メモリ上のログをディスク装置に移動させ、保存させることを特徴とするログ取得方法。

【請求項２０】　請求項１乃至１９のいずれか１つに記載のログ取得方法をコンピュータによって実現させるための制御プログラムを格納した記憶媒体。

【請求項２１】　請求項１乃至１９のいずれか１つに記載のログ取得方法をコンピュータによって実現させるための制御プログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、複数にモジュール分けされたソフトウェアの処理ログを取得するための技術に関するものである。

【０００２】

【従来の技術】

従来より、再現率の低いソフトウェアの障害に対しては、ソフトウェアの処理ログを取得し、当該処理ログを解析することによって、障害の原因をつきとめ、その対策を講じてきた。

【０００３】

【発明が解決しようとする課題】

しかし、上記従来の処理ログの取得には以下のような問題点がある。

（１）ユーザの動作環境でもログを取得しつづけるためには、ソフトウェアのモジュール自体に手を加え、処理ログ取得ルーチンを追加しなければならず、処理ログ取得のための作業負荷が大きかった。

（２）処理ログ取得はモジュール毎に行うため、生成されたログはモジュール単位のものとなってしまい、ソフトウェア全体の処理を、完全に時間順のログとして取得するのが困難である。このため、全体の処理ログとしての見通しが悪く、ログを解析して障害の原因を発見するまでのプロセスに工数がかかっていた。

【０００４】

本発明は、上記課題を鑑みてなされたものであり、複数にモジュール分けされたソフトウェアの処理ログを容易に取得でき、かつ、ソフトウェアの障害の原因の解析のための工数を削減することが可能なログ取得方法、ならびに該方法をコンピュータによって実現させるためのプログラム、および該プログラムを格納した記憶媒体を提供することを目的とする。

【０００５】

【課題を解決するための手段】

上記の目的を達成するために本発明に係るログ取得方法は以下のような構成を備える。即ち、

所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記選択する工程において選択された関数を呼び出す際の所定の情報をログとして記録する工程と、

前記選択する工程において選択された関数の実行結果を受け取った際の所定の情報をログとして記録する工程とを備える。

【０００６】

【発明の実施の形態】

【第１の実施形態】

本実施形態は、あるモジュールから別のモジュール内に存在する関数の呼び出しが行われる際の仕組みである、メモリに保持されたインポート関数アドレス、または仮想関数アドレステーブル（Ｖｉｒｔｕａｌ　Ａｄｄｒｅｓｓ　Ｔａｂｌｅ）を利用して、モジュール間の関数呼び出しをフックしてログに記録することで、ソフトウェアのモジュール自体に手を加えることなく、ソフトウエア全体の処理を、時間順のログとして取得することを可能にするものである。以下に具体的に説明する。

【０００７】

＜システム構成＞

図１は、本発明の各実施形態にかかるログ取得方法を実現するコンピュータ（ソフトウェア評価システム）の構成をあらわす図である。説明を簡略化するために、本実施形態では、本ソフトウェア評価システムが１台のＰＣ内部に構築されるものとするが、本発明のログ取得方法の特徴は１台のＰＣ内部に構築されるか、あるいは複数のＰＣにネットワークシステムとして構築されるかによらず有効である。

【０００８】

本ソフトウェア評価システムを搭載するコンピュータには、ＣＰＵ１、チップセット２、ＲＡＭ３、ハードディスクコントローラ４、ディスプレイコントローラ５、ハードディスクドライブ６、ＣＤ－ＲＯＭドライブ７、ディスプレイ８が搭載されている。また、ＣＰＵとチップセットを繋ぐ信号線１１、チップセット２とＲＡＭ３とを繋ぐ信号線１２、チップセット２と各種周辺機器とを繋ぐ周辺機器バス１３、ハードディスクコントローラ４とハードディスクドライブ６とを繋ぐ信号線１４、ハードディスクコントローラ４とＣＤ－ＲＯＭドライブ７とを繋ぐ信号線１５、ディスプレイコントローラ５とディスプレイ８とを繋ぐ信号線１６が搭載されている。

【０００９】

＜関数処理に対する処理ログ取得＞

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムを説明するために、まず図２によって、複数のモジュールに分かれたソフトウェアが、通常の状態でどのようにメモリにロードされるかを説明する。

【００１０】

通常、複数のモジュールに分かれたソフトウェアは、全体の制御を行う実行ファイルＥＸＥ(２３)と、モジュールとして存在しＥＸＥの補完的な役割を担うダイナミックリンクライブラリＤＬＬ(２７)とに分かれており、メモリにはＥＸＥとＤＬＬの両方がロードされる。ＥＸＥはコードセグメント(２８)とデータセグメント(２９)、そしてインポート関数アドレステーブル(２２)から成っている。更に、インポート関数アドレステーブルは、関数の所属するＤＬＬによって分かれており(２１，　２４)、ＤＬＬごとにそれぞれの関数がロードされたアドレスが書かれている(３０～３５)。ＤＬＬの関数の実体は、ＤＬＬごとに分かれて(２５，　２６)ロードされ、それぞれの関数は該当するＤＬＬの一部としてロードされる(３６～４１)。この図では、１本のＥＸＥがA.ＤＬＬ及びB.ＤＬＬの２つのダイナミックリンクライブラリ内の関数を使用している例を示しており、実際に使用される関数はＦｕｎｃ　ＡＡ，　Ｆｕｎｃ　ＡＢ，　Ｆｕｎｃ　ＡＣ，　Ｆｕｎｃ　ＢＡ，　Ｆｕｎｃ　ＢＢ，　Ｆｕｎｃ　ＢＣの６個となっている。

【００１１】

ＥＸＥのコードセグメント内にあるコードが関数Ｆｕｎｃ　ＡＡを呼び出す場合には、まずインポート関数アドレステーブル内に書かれたＦｕｎｃ　ＡＡのアドレス(３０)が読み込まれる。ここには実際にはA.ＤＬＬの一部として読み込まれたＦｕｎｃ　ＡＡコード(３６)のアドレスが書かれており、そのアドレスをコールすることによって、ＥＸＥのコードはA.ＤＬＬのＦｕｎｃ　ＡＡを呼び出すことができる。

【００１２】

図３は、本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのメモリ構成をあらわす図であり、図２とは、ログ取得用のコードに対してＩＡＴ　Ｐａｔｃｈ(Ｉｍｐｏｒｔ　Ａｄｄｒｅｓｓ　Ｔａｂｌｅ

Ｐａｔｃｈ)という手法を用いて、関数呼び出しをリダイレクトしているという点で異なっている。

【００１３】

ログ取得が開始されると、メモリ内にはＩＡＴ　Ｐａｔｃｈ用のＤＬＬであるＣ.ＤＬＬ(５８)がロードされる。Ｃ.ＤＬＬはインポート関数アドレステーブル(５２)内に書かれた関数のアドレスを、Ｃ.ＤＬＬ内のログ取得コードであるＦｕｎｃ　ＣＡＡ，Ｆｕｎｃ　ＣＡＢ，Ｆｕｎｃ　ＣＡＣ，Ｆｕｎｃ　ＣＢＡ，Ｆｕｎｃ　ＣＢＢ，Ｆｕｎｃ　ＣＢＣのアドレスに書き換える(６１～６６)。Ｃ.ＤＬＬ内のＦｕｎｃ　ＣＡＡ，Ｆｕｎｃ　ＣＡＢ，Ｆｕｎｃ　ＣＡＣ，Ｆｕｎｃ　ＣＢＡ，Ｆｕｎｃ　ＣＢＢ，Ｆｕｎｃ　ＣＢＣのコード(７３～７８)は、ログを記録すると共に、元々の関数呼び出しを受けるべくメモリにロードされている、該当する関数であるＦｕｎｃ　ＡＡ，Ｆｕｎｃ　ＡＢ，Ｆｕｎｃ　ＡＣ，Ｆｕｎｃ　ＢＡ，Ｆｕｎｃ　ＢＢ，Ｆｕｎｃ　ＢＣ(６７～７２)を呼び出す。

【００１４】

図４Ａは、図３におけるＩＡＴ　Ｐａｔｃｈの処理をあらわす図、図４Ｂはログ取得処理の流れを示すフローチャートである。説明を簡略化するために、この図ではＥＸＥがＡ.ＤＬＬ内のＦｕｎｃ　ＡＡを呼び出す際に、ＩＡＴ　Ｐａｔｃｈによるログ取得コードがどのように動作するかの例をあらわしている。

【００１５】

ＥＸＥ(図４Ａの９１)がＦｕｎｃ　ＡＡをコールすると(図４Ａの９４)、Ｃ.ＤＬＬ内にあるログ取得コードがＤＬＬ名／関数名をメモリに保存し（図４ＢのステップＳ４０２）、呼び出し時刻をメモリに保存し、呼び出し時のパラメータをメモリに保存し、呼び出し時のポインタパラメータの指すメモリ内容を、別メモリに保存する(図４Ａの９５、図４ＢのステップＳ４０３）。その後Ｃ.ＤＬＬは本来呼び出されるはずであった、Ａ.ＤＬＬ(図４Ａの９３)内のＦｕｎｃ　ＡＡをコールする(図４Ａの９６、図４ＢのステップＳ４０４）。Ａ.ＤＬＬのＦｕｎｃ　ＡＡ処理(図４Ａの９７)が終了し、Ｃ.ＤＬＬに制御がリターンすると(図４Ａの９８)、Ｃ.ＤＬＬはリターン時の時刻をメモリに保存し、戻り値をメモリ

に保存し、リターン時にポインタパラメータが指すメモリ内容を、別メモリに保存する(図４Ａの９９)。その後、Ｃ.ＤＬＬは保存したログ情報をファイルに書き込み(図４Ａの１００、図４Ｂのステップ Ｓ４０５)、あたかもＡ.ＤＬＬのＦｕｎｃ　ＡＡが通常通りに終了したかのように、ＥＸＥにリターンする(１０１)。

【００１６】

図５は、本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの内部構成をあらわす図である。通常は実行形式のＥＸＥ(１１３)が、ＤＬＬ－１(１１６)やＤＬＬ－２(１１７)内の関数を呼び出すが、ここではＡＰＩトレーサと呼ばれるログ取得コードを埋め込み(１１４)、処理ログを生成している(１１５)。ＡＰＩトレーサは、ＤＬＬ－１やＤＬＬ－２の関数定義を記述したファイル(１１１)と、どのＤＬＬのどの関数のインポート関数テーブルを書き換えてログを取得するかの設定シナリオ(トレースシナリオ１１２)を元に動作する。

【００１７】

＜メソッド処理に対する処理ログ取得＞

次に、本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムにおいて、実行ファイルＥＸＥ(１１８)がＣＯＭ（Ｃｏｍｐｏｎｅｎｔ　Ｏｂｊｅｃｔ　Ｍｏｄｅｌ）サーバでエクスポートされているインターフェースのインスタンス作成時に、どのようにメモリにロードされるかを説明するために、まず、図６によって、通常の状態でどのようにメモリにロードされるかを説明する。

【００１８】

通常、インターフェースのインスタンス作成を行うと、ＣＯＭサーバ内で、要求されたインターフェース(１２１，　１２２)と、そのメソッド(：オブジェクト指向プログラミングにおいて、オブジェクトの実行する手続きを記述したプログラム、１３０～１３５)が作成され、それらは、メモリ上に両方がロードされる。ここで、ｖｉｒｔｕａｌ　ａｄｄｒｅｓｓ　ｔａｂｌｅ（仮想アドレステーブル）は作成された各インターフェース毎に作られ(１１８，　１２０)、作成要求

を行ったＥＸＥに渡される。このｖｉｒｔｕａｌ　ａｄｄｒｅｓ　ｔａｂｌｅには各メソッドの作成されたアドレスが書かれている(１２４～１２９)。ＥＸＥはこれら情報を利用し、各インターフェースに対して呼び出しを行う。この図では、１本のＥＸＥがＩｎｔｅｒｆａｃｅ　Ａ及びＩｎｔｅｒｆａｃｅ　Ｂの２つのインターフェースのインスタンスを作成しており、そのインターフェース内部のメソッドを使用している例を示しており、実際に使用されているメソッドは、Ｍｅｔｈｏｄ　ＡＡ，Ｍｅｔｈｏｄ　ＡＢ，Ｍｅｔｈｏｄ　ＡＣ，Ｍｅｔｈｏｄ　ＢＡ，Ｍｅｔｈｏｄ　ＢＢ，Ｍｅｔｈｏｄ　ＢＣとなっている。

【００１９】

ＥＸＥのコードが関数Ｍｅｔｈｏｄ　ＡＡを呼び出す場合には、まずｖｉｒｔｕａｌ　ａｄｄｒｅｓｓ　ｔａｂｌｅ内に書かれたＭｅｔｈｏｄ　ＡＡのアドレス(１２４)が読み込まれる。ここには実際にはＣＯＭサーバのＩｎｔｅｒｆａｃｅ　Ａの一部として作成されたＭｅｔｈｏｄ　ＡＡコード(１３０)のアドレスが書かれており、そのアドレスをコールすることによって、ＥＸＥのコードはＩｎｔｅｒｆａｃｅ　ＡのＭｅｔｈｏｄ　ＡＡを呼び出すことができる。

【００２０】

図７は、本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのメモリ構成をあらわす図であり、図６とは、ログ取得用のコードに対してＶＴａｂｌｅ　Ｐａｔｃｈ(ｖｉｒｔｕａｌ　ａｄｄｒｅｓｓ　ｔａｂｌｅ　Ｐａｔｃｈ)という手法を用いて、メソッド呼び出しをリダイレクトしているという点で異なっている。

【００２１】

ログ取得が開始されると、メモリ内にはＶＴａｂｌｅ　Ｐａｔｃｈ用のＤＬＬ(１４３)がロードされる。このＤＬＬはｖｉｒｔｕａｌ　ａｄｄｒｅｓｓ　ｔａｂｌｅ(１３６，　１３８)内に書かれたメソッドのアドレスを、ＤＬＬ内のログ取得コードであるＭｅｔｈｏｄ　Ａ’Ａ，Ｍｅｔｈｏｄ　Ａ’Ｂ，Ｍｅｔｈｏｄ　Ａ’Ｃ，Ｍｅｔｈｏｄ　Ｂ’Ａ，Ｍｅｔｈｏｄ　Ｂ’Ｂ，Ｍｅｔｈｏｄ　Ｂ’Ｃのアドレスに書き換える(１４５～１５０)。ＤＬＬ内のＭｅｔｈｏｄ　Ａ’Ａ，Ｍｅｔｈｏｄ　Ａ’Ｂ，Ｍｅｔｈｏｄ　Ａ’Ｃ，Ｍｅｔｈｏｄ　Ｂ’Ａ，Ｍｅ

ｔｈｏｄ　Ｂ'Ｂ，　Ｍｅｔｈｏｄ　Ｂ'Ｃのコード(１５７～１６２)は、ログを記録すると共に、元々のメソッド呼び出しを受けるべくメモリにロードされている、該当するメソッドであるＭｅｔｈｏｄ　ＡＡ，　Ｍｅｔｈｏｄ　ＡＢ，　Ｍｅｔｈｏｄ　ＡＣ，　Ｍｅｔｈｏｄ　ＢＡ，　Ｍｅｔｈｏｄ　ＢＢ，　Ｍｅｔｈｏｄ　ＢＣ(１５７～１６２)を呼び出す。

【００２２】

図８Ａは、図７におけるＶＴａｂｌｅ　Ｐａｔｃｈの処理をあらわす図、図８Ｂはログ取得処理の流れを示すフローチャートである。説明を簡略化するために、この図ではＥＸＥがＣＯＭサーバ内のＩｎｔｅｒｆａｃｅ　ＡのＭｅｔｈｏｄ　ＡＡを呼び出す際に、ＶＴａｂｌｅ　Ｐａｔｃｈによるログ取得コードがどのように動作するかの例をあらわしている。

【００２３】

ＥＸＥ(図８Ａの１６３)がＭｅｔｈｏｄ　ＡＡをコールすると(図８Ａの１６６)、ＤＬＬ内にあるログ取得コードがモジュール名／インターフェース名／メソッド名をメモリに保存し（図８Ｂのステップ Ｓ８０２）、呼び出し時刻をメモリに保存し、呼び出し時のパラメータをメモリに保存し、呼び出し時のポインタパラメータの指すメモリ内容を、別メモリに保存する(図８Ａの１６７、図８ＢのステップＳ８０３）)。その後ＤＬＬは本来呼び出されるはずであった、ＣＯＭサーバ(図８Ａの１６５)内のＭｅｔｈｏｄ　ＡＡをコールする(図８Ａの１６８、図８ＢのステップＳ８０４）)。ＣＯＭサーバのＭｅｔｈｏｄ　ＡＡ処理(図８Ａの１６９)が終了し、ＤＬＬに制御がリターンすると(図８Ａの１７０)、ＤＬＬはリターン時の時刻をメモリに保存し、戻り値をメモリに保存し、リターン時にポインタパラメータが指すメモリ内容を、別メモリに保存する(図８Ａの１７１)。その後、ＤＬＬは保存したログ情報をファイルに書き込み(図８Ａの１７２、図８Ｂのステップ Ｓ８０５)、あたかもＣＯＭサーバのＭｅｔｈｏｄ　ＡＡが通常通りに終了したかのように、ＥＸＥにリターンする(図８Ａの１７３)。

【００２４】

図９は、本発明の第１の実施形態にかかるソフトウェア評価システムの内部構成をあらわす図である。通常は実行形式のＥＸＥ(１７６)が、ＣＯＭサーバ１(

１７９)やＣＯＭサーバ２(１８０)内のメソッドを呼び出すが、ここではＡＰＩトレーサと呼ばれるログ取得コードを埋め込み(１７７)、処理ログを生成している(１７８)。ＡＰＩトレーサは、ＣＯＭサーバ１(１７９)やＣＯＭサーバ２の関数定義を記述したファイル(１７４)と、どのＣＯＭサーバのどのインターフェースのどのメソッドのｖｉｒｔｕａｌ　ａｄｄｒｅｓｓ　ｔａｂｌｅを書き換えてログを取得するかの設定シナリオ(１７５)を元に動作する。

【００２５】

＜ユーザインターフェース＞

図１０は、本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、ログ取得を開始するための関数／メソッドを設定するためのユーザインターフェースである。

【００２６】

ユーザインターフェースには、ログ取得対象となっているモジュール／インターフェース(１８１)から、どのモジュール／インターフェースを設定するかを選択するドロップダウンリスト(１８３，１８４)と、そこで選択したモジュール／インターフェースでエクスポートされている関数／メソッド(１８２)から、どの関数／メソッドを設定するかを選択するドロップダウンリスト(１８５，１８６)とを備える。

【００２７】

図１１は、本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、図１０で設定した内容に従い、ログを取得する場合の流れを示す図である。

【００２８】

処理が開始されると(ステップＳ１)、ログ取得コードは呼び出された関数／メソッドがログ取得開始トリガーとして設定されているか判別する(ステップＳ２)。ログ取得開始トリガーと一致した場合には、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をＨＤＤに保存する(ステップＳ３)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存し(ステップＳ４)、元の関数／メソッドを

呼び出す(ステップＳ５)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存する(ステップＳ６)。この処理はユーザから終了の命令があるまで(ステップＳ７)続行されるが、その際には、ログ取得開始トリガーの判別を行わない。

【００２９】

以上の説明から明らかなように、本実施形態によれば、複数にモジュール分けされたソフトウエアの処理ログの取得において、モジュール自体に変更を加えることなく、モジュール内に用意された任意の関数／メソッドから処理ログを容易に取得することが可能となり、ソフトウェアにおける障害の原因の解析のための工数を削減する効果を与える。さらにログ取得を行う関数／メソッドを任意に選択できるようにすることで、ログの解析が容易になるという効果がもたらされる。

【００３０】

【第２の実施形態】

上記第１の実施形態では、ログを取得する関数／メソッドをユーザが任意に選択することが可能なユーザインターフェースについて説明したが、本実施形態では、ログ取得を停止する関数／メソッドを任意に選択可能なユーザインターフェースについて説明する。

【００３１】

図１２は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、ログ取得を停止するための関数／メソッドを設定するためのユーザインターフェースである。

【００３２】

ユーザインターフェースには、ログ取得対象となっているモジュール／インターフェース(１８７)から、どのモジュール／インターフェースを設定するかを選択するドロップダウンリスト(１８９，１９０)と、そこで選択したモジュール／インターフェースでエクスポートされている関数／メソッド(１８８)から、どの関数／メソッドを設定するかを選択するドロップダウンリスト(１９１，１９２)

とを備える。

【００３３】

図１３は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、図１２で設定した内容に従い、ログを取得する場合の流れを示す図である。

【００３４】

処理が開始されると(ステップＳ９)、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をＨＤＤに保存する(ステップＳ１０)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存し(ステップＳ１１)、元の関数／メソッドを呼び出す(ステップＳ１２)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存する(ステップＳ１３)。その後呼び出された関数／メソッドがログ取得停止トリガーとして設定されているか判別する(ステップＳ１４)。ログ取得停止トリガーと一致した場合には、ログ取得処理は終了する(ステップＳ１６)。また、ログ取得停止トリガーと一致しない場合でも、この処理はユーザから終了の命令があると(ステップＳ１５)終了される。

【００３５】

これにより、任意の関数／メソッドのログ取得を停止することができることとなり、ユーザの求めるログのみを取得できるため、ログの解析が容易になるという効果がもたらされる。

【００３６】

【第３の実施形態】

上記第１および第２の実施形態において述べたユーザインタフェースでは、ユーザが選択した任意の関数／メソッドでログの取得開始／停止をユーザが任意に選択することとしたが、ユーザが選択した任意の関数／メソッドが、エラーで終了した場合のみ、ログの取得開始／停止をするように設定することも可能である。

【００３７】

図１４は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、ログ取得を開始／停止するための関数／メソッドを設定するためのユーザインターフェースに、エラーで終了した場合のみトリガー機能（ログ取得開始／停止機能）を使用する設定を追加したものである。本ユーザインターフェースは、図１０及び、図１２双方に適応することが可能である。

【００３８】

ユーザインターフェースには、ログ取得対象となっているモジュール／インターフェース(１９３)から、どのモジュール／インターフェースを設定するかを選択するドロップダウンリスト(１９５，１９６)と、そこで選択したモジュール／インターフェースでエクスポートされている関数／メソッド(１９４)から、どの関数／メソッドを設定するかを選択するドロップダウンリスト(１９７，１９８)と、関数／メソッドがエラーで終了した場合のみトリガー機能を有効にする為のチェックボックスとを備える。

【００３９】

図１５は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、関数／メソッドのエラー定義の内容を示したものである。本実施形態においてはエラー定義がファイルとなっており、ファイルには各関数／メソッドのパラメータ及び戻り値とそのエラー条件等が定義され記されている。

【００４０】

図１６は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図１４で設定した内容に従い、ログを取得する場合の流れを示す図であり、ログ取得開始と、エラー時のみトリガーを使用する機能を併用して設定した場合である。

【００４１】

処理が開始されると(ステップＳ１７)、ログ取得コードは呼び出された関数／メソッドがログ取得開始トリガーとして設定されているか判別する(ステップＳ１８)。ログ取得開始トリガーとして設定されている場合には、モジュール名、インターフェース名、関数／メソッド名をメモリに一時的に保存する(ステップＳ１９)。

【００４２】

次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をメモリに一時的に保存し(ステップＳ２０)、元の関数／メソッドを呼び出す(ステップＳ２１)。メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をメモリに一時的に保存する(ステップＳ２２)。

【００４３】

次にログ取得開始トリガーがエラーでのみ使用するかどうかを判別し(ステップＳ２３)、そうであれば、関数／メソッドがエラーであるかを判別する(ステップＳ２４)。もし、エラーでない場合には、一時メモリに保存したログを破棄し(ステップＳ２５)、処理のはじめに戻る。関数／メソッドがエラーであった場合には、一時メモリに保存したログをＨＤＤに保存し(ステップＳ２６)、通常のログ取得処理を引き続き行う(ステップＳ２７)。この処理はユーザから終了の命令があるまで(ステップＳ２８)続行される。

【００４４】

図１７は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図１６のステップＳ２７で示した通常のログ取得処理の詳細の流れを示す図である。

【００４５】

処理が開始されると(ステップＳ３０)、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をＨＤＤに保存する(ステップＳ３１)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存し(ステップＳ３２)、元の関数／メソッドを呼び出す(ステップＳ３３)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存する(ステップＳ３４)。

【００４６】

図１８は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図１４で設定した内容に従い、ログを取得する場合の流れを示す図であ

り、ログ取得停止と、エラー時のみトリガーを使用する機能とを併用して設定した場合である。

【００４７】

処理が開始されると(ステップＳ４０)、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をＨＤＤに保存する(ステップＳ４１)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存し(ステップＳ４２)、元の関数／メソッドを呼び出す(ステップＳ４３)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存する(ステップＳ３４)。その後呼び出された関数／メソッドがログ取得停止トリガーとして設定されているか判別する(ステップＳ４５)。ログ取得停止トリガーと一致した場合には、次にログ取得停止トリガーがエラーでのみ使用するかどうかを判別し(ステップＳ４６)、そうであれば、関数／メソッドがエラーであるかを判別する(ステップＳ４７)。もし、エラーである場合は、ログ取得処理は終了する(ステップＳ４８)。また、この処理はユーザから終了の命令があると(ステップＳ４９)終了される。

【００４８】

これにより、任意の関数／メソッドでエラーが発生した場合からログの取得を開始したり、停止したりすることが可能ととなり、ユーザの求めるログを取得することが可能になり、ログの解析が容易になるという効果がもたらされる。

【００４９】

【第４の実施形態】

上記第１乃至第３の実施形態におけるユーザインターフェースは、選択可能な関数／メソッドが所定の順序で配列されているにすぎなかったが、インターフェース、メソッドの関係が容易に把握できるようなツリー表示としてもよい。

【００５０】

図１９は、本発明の第４の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、インターフェースとメソッドのツリー表示を行うユーザインターフェースに関するものである。

【００５１】

ユーザインターフェースには、インターフェース及びメソッドをツリー表示する為のビュー(２００)が備えられている。ユーザがインターフェースＩｎｔｅｒｆａｃｅＡ(２０１)をチェックすることで、そのインターフェース内の全てのメソッドＭｅｔｈｏｄＡＡ，ＭｅｔｈｏｄＡＢ，ＭｅｔｈｏｄＡＣ(２０２～２０４)は全て選択され、ログの取得対象となる。また、インターフェースＩｎｔｅｒｆａｃｅＢ(２０５)のチェックを外すことで、そのインターフェース内の全てのメソッドＭｅｔｈｏｄＢＡ，ＭｅｔｈｏｄＢＢ，ＭｅｔｈｏｄＢＣ(２０５～２０８)は全て選択解除され、ログの取得対象外となる。

【００５２】

図２０は、本実施形態にかかるソフトウェア評価システムの、図１９のようにログ取得対象を選択した場合の、ログを取得する際の処理の流れを示す図である。

【００５３】

処理が開始されると(ステップＳ５１)、ログ取得コードはインターフェースの何らかのメソッド呼び出しが行われるたびに、そのメソッドのインターフェースが、ログ取得対象となっているかを判別する(ステップＳ５２)。ログ取得対象となっていた場合には、モジュール名とインターフェース、メソッド名をＨＤＤに保存する(ステップＳ５３)。

【００５４】

次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存し(ステップＳ５４)、元のメソッドを呼び出す(ステップＳ５５)。メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存する(ステップＳ５６)。この処理はユーザから終了の命令があるまで(ステップＳ５７)続行される。

【００５５】

これにより、インターフェース、メソッドの関係を更に簡便に知ることができ、また、ログを取得したいインターフェースを容易に選択可能となり、ユーザの

、求めるログを取得する方法が簡単になるという効果がもたらされる。

【００５６】

【第５の実施形態】

上記第１乃至第３の実施形態では、任意のメソッドを選択するにあたり、インターフェース単位で選択を行うこととしたが、さらに、メソッド単位で選択を行うことも可能である。

【００５７】

図２１は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、インターフェースとメソッドのツリー表示を行うユーザインターフェースに関するものであり、図１９と同様であるが、選択方法が異なっている。

【００５８】

ユーザインターフェースには、インターフェース及びメソッドをツリー表示する為のビュー(２０９)が備えられている。ユーザがメソッドＭｅｔｈｏｄＡＡ(２１１)，ＭｅｔｈｏｄＡＣ(２１３)，ＭｅｔｈｏｄＢＡ(２１５)をチェックし、ＭｅｔｈｏｄＡＢ(２１２)，ＭｅｔｈｏｄＢＢ(２１６)，ＭｅｔｈｏｄＢＣ(２１７)のチェックを外すことで、インターフェースＩｎｔｅｒｆａｃｅＡ(２１０)，ＩｎｔｅｒｆａｃｅＢ(２１４)内の全てのメソッドではなく、各インターフェースのチェックしたメソッドのみをログの取得対象とすることが可能となる。

【００５９】

図２２は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図２１のようにログ取得対象を選択した場合の、ログを取得する際の処理の流れを示す図である。

【００６０】

処理が開始されると(ステップＳ５９)、ログ取得コードはインターフェースの何らかのメソッド呼び出しが行われるたびに、そのインターフェースのメソッドが、ログ取得対象となっているかを判別する(ステップＳ６０)。ログ取得対象となっていた場合には、モジュール名とインターフェース、メソッド名をＨＤＤに保存する(ステップＳ６１)。

【００６１】

次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存し(ステップＳ６２)、元のメソッドを呼び出す(ステップＳ６３)。メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をＨＤＤに保存する(ステップＳ６４)。この処理はユーザから終了の命令があるまで(ステップＳ６５)続行される。

【００６２】

これにより、インターフェース毎という大きな範囲ではなく、ログを取得したいメソッドをメソッド単位でそれぞれ容易に選択可能となり、ユーザの求めるログを簡単に取得することができるという効果がもたらされる。

【００６３】

【第６の実施形態】

上記各実施形態においては、取得したログをＨＤＤの任意の場所に保存していたが、ログの解析を容易に行うことができるよう、日付ごとに保存するようにしてもよい。

【００６４】

図２３は、日付毎にログを分割保存する場合の処理の流れを示す図である。

【００６５】

処理が開始されると(ステップＳ７１)、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をメモリに保存する(ステップＳ７２)。

【００６６】

次にログ取得コードは、呼び出し時の日付・時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をメモリに保存し(ステップＳ７３)、元の関数／メソッドを呼び出す(ステップＳ７４)。関数／メソッドからリターンすると、ログ取得コードはリターン時の日付・時刻、戻り値及びポインタ・パラメータの指すメモリの内容をメモリに保存する(ステップＳ７５)。その後呼び出された関数／メソッドのリターン時の日付が、前回に保存したログのリターン時の日付と異なるかを判別する(ステップＳ７６)。日付が異なる場合には、新規のログファイ

ルを作成し、ログをそのファイルに保存する(ステップＳ７７)、同じ場合には、従来のログファイルにログを保存する(ステップＳ７８)。この処理はユーザから終了の命令があると(ステップＳ７９)終了される（ステップＳ８０）。

【００６７】

これにより、ユーザは求めるログを日付毎に取得することができ、ログの解析が容易になるという効果がもたらされる。

【００６８】

【第７の実施形態】

上記第６の実施形態では、日付ごとにログを保存する場合について述べたが、サイズ毎もしくは個数毎にログを分割して保存するようにしてもよい。

【００６９】

図２４は、サイズ毎もしくは個数毎にログを分割保存する場合の処理の流れを示す図である。

【００７０】

処理が開始されると(ステップＳ７１)、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をメモリに保存する(ステップＳ７２)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をメモリに保存し(ステップＳ７３)、元の関数／メソッドを呼び出す(ステップＳ７４)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をメモリに保存する(ステップＳ７５)。その後新規のログデータを従来のログファイルに保存した場合に、ファイルのサイズが一定値を超えるかもしくは一定の個数のログを超えるかどうかを判別する(ステップＳ７６)。

【００７１】

ある一定サイズを超えてしまう場合もしくはある一定の個数を超えてしまう場合には、新規のログファイルを作成し、ログをそのファイルに保存する(ステップＳ７７)、同じ場合には、従来のログファイルにログを保存する(ステップＳ１８)。新規のログファイルを作成した場合には、リングバッファを使用する場合で且つ、作成したログファイルの数が２個より多いかどうかを判別する(ステッ

プＳ７９)。もし、条件に当てはまる場合には、最も古いログファイルを削除する(ステップＳ９０)。この処理はユーザから終了の命令があると(ステップＳ９１)終了される（ステップＳ９２）。

【００７２】

これにより、ユーザは複数作成される各ログを一定サイズ以内で取得することや、一定のログの個数毎で取得ができ、取り扱いが容易になる効果がもたらされる。また、リングバッファを使用することで、本ソフトウェア評価システムがＰＣのリソースに対して与える負荷を一定に押さえる事が可能になり、安定したログの取得を行えるようになる。

【００７３】

【第８の実施形態】

図２５は、取得したログを一定数だけメモリに保存する場合のメモリの概要図である。

【００７４】

一定数のログを保存するためのログ格納メモリ領域がｎ個存在し(２１８)、各ログ格納メモリ領域には、１回分の関数／メソッドのログが格納され、その情報はモジュール名、インターフェース名、関数／メソッド名、呼び出し時の時刻、呼び出し時のパラメータデータ、終了時の時刻、終了時のパラメータデータ、戻り値データ等が格納されており(２１９)、可変サイズとなる。このメモリ領域には、ログ格納メモリ領域１から順にログデータは格納されていき、ログ格納メモリ領域ｎまで、使用し終わると、再度、ログ格納メモリ領域１から上書きしてログを格納する。

【００７５】

図２６は、取得したログを一定数だけメモリに保存する場合のログ取得の流れを示す図である。

【００７６】

処理が開始されると(ステップＳ９３)、ログ格納メモリ領域の場所を示す変数ｘを１に初期化する(ステップＳ９４)。そして、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をログ格納メモリ領域ｘに保存する

(ステップＳ９５)。

【００７７】

次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をログ格納メモリ領域ｘに保存し(ステップＳ９６)、元の関数／メソッドを呼び出す(ステップＳ９７)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をログ格納メモリ領域ｘに保存する(ステップＳ９８)。その後ログ格納メモリ領域の場所を示す変数ｘに１を加算し(ステップＳ９９)、ｘがログ格納メモリ領域数のｎより大きいかを判別する(ステップＳ１００)。

【００７８】

ｘがｎより大きい場合にはｘに１を代入し、ログ格納メモリ領域の先頭から再度使用するように設定する(ステップＳ１０１)。その後、リングバッファを使用するかを判別し(ステップＳ１０２)、使用しない場合には、メモリ上に有る、全てのログデータをログファイルに保存し(ステップＳ１０３)、メモリ上のログデータを全て削除する(ステップＳ１０４)。この処理はユーザから終了の命令があると(ステップＳ１０５)終了される（ステップＳ１０６）。

【００７９】

これにより、メモリの使用量をある程度の値でとどめることが可能となり、本ソフトウェア評価システムがＰＣのリソースに対して与える負荷を抑えることが可能になり、安定したログの取得を行えるようになる。

【００８０】

【他の実施形態】

なお、本発明は、複数の機器（例えばホストコンピュータ，インタフェイス機器、リーダ、プリンタなど）から構成されるシステムに適用しても、一つの機器からなる装置（例えば、複写機、ファクシミリ装置など）に適用してもよい。

【００８１】

また、本発明の目的は、前述した実施形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体を、システムあるいは装置に供給し、そのシステムあるいは装置のコンピュータ（またはＣＰＵやＭＰＵ）が記憶媒体に格納

されたプログラムコードを読出し実行することによっても、達成されることは言うまでもない。

【００８２】

この場合、記憶媒体から読出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。

【００８３】

プログラムコードを供給するための記憶媒体としては、例えば、フロッピ（登録商標）ディスク、ハードディスク、光ディスク、光磁気ディスク、ＣＤ－ＲＯＭ、ＣＤ－Ｒ、磁気テープ、不揮発性のメモリカード、ＲＯＭなどを用いることができる。

【００８４】

また、コンピュータが読出したプログラムコードを実行することにより、前述した実施形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼働しているＯＳ（オペレーティングシステム）などが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【００８５】

さらに、記憶媒体から読出されたプログラムコードが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書込まれた後、そのプログラムコードの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるＣＰＵなどが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【００８６】

【発明の効果】

以上説明したように本発明によれば、複数にモジュール分けされたソフトウェアの処理ログを容易に取得でき、かつ、ソフトウェアの障害の原因の解析のための工数を削減することが可能となる。

【図面の簡単な説明】

【図１】

本発明の第１の実施形態にかかるログ取得方法を実現するコンピュータ（ソフトウェア評価システム）の構成をあらわす図である。

【図２】

本発明の第１の実施形態にかかるログ取得方法を説明するためにあらわした、関数ロード時の通常のメモリ構成をあらわす図である。

【図３】

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのＩＡＴ　Ｐａｔｃｈ使用時のメモリ構成をあらわす図である。

【図４Ａ】

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのＩＡＴ　Ｐａｔｃｈを使用時の状態を示す図である。

【図４Ｂ】

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのログ取得処理の流れを示す図である。

【図５】

本発明の第１の実施形態にかかるソフトウェア評価システムのＩＡＴ　Ｐａｔｃｈを使用時の内部構成をあらわす図である。

【図６】

本発明の第１の実施形態にかかるログ取得方法を説明するためにあらわした、ＣＯＭサーバのインターフェースのインスタンス作成時の通常のメモリ構成をあらわす図である。

【図７】

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのＶＴａｂｌｅ　Ｐａｔｃｈ使用時のメモリ構成をあらわす図である。

【図８Ａ】

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのＶＴａｂｌｅ　Ｐａｔｃｈを使用時の状態を示す図である。

【図８Ｂ】

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのログ取得処理の流れを示す図である。

【図９】

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、内部構成をあらわす図である。

【図１０】

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、ログ取得を開始するための関数／メソッドを設定するためのユーザインターフェースを示す図である。

【図１１】

本発明の第１の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、図１０で設定した内容に従い、ログを取得する場合の流れを示す図である。

【図１２】

本発明の第２に実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、ログ取得を停止するための関数／メソッドを設定するためのユーザインターフェースを示す図である。

【図１３】

本発明の第２の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、図１２で設定した内容に従い、ログを取得する場合の流れを示す図である。

【図１４】

本発明の第３の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、ログ取得を開始／停止するための関数／メソッドを設定するためのユーザインターフェースに、エラーで終了した場合のみトリガー機能を使用する設定を追加したユーザインターフェースを示す図である。

【図１５】

本発明の第３の実施形態にかかるログ取得方法を実現するソフトウェア評価シ

ステムの、関数／メソッドのエラー定義の内容を示した図である。

【図１６】

本発明の第３の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図１４で設定した内容に従い、ログを取得する場合の流れを示す図である。

【図１７】

本発明の第３の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図１６のステップＳ２７で示した通常のログ取得処理の詳細の流れを示す図である。

【図１８】

本発明の第３の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図１４で設定した内容に従い、ログを取得する場合の流れを示す図である。

【図１９】

本発明の第４の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、インターフェースとメソッドのツリー表示を行うユーザインターフェースを示す図である。

【図２０】

本発明の第４の実施形態にかかるソフトウェア評価システムの、図１９のようにログ取得対象を選択した場合の、ログを取得する際の処理の流れを示す図である。

【図２１】

本発明の第５の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、インターフェースとメソッドのツリー表示を行うユーザインターフェースを示す図である。

【図２２】

本発明の第５の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図２１のようにログ取得対象を選択した場合の、ログを取得する際の処理の流れを示す図である。

【図２３】

本発明の第６の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、日付毎にログを分割保存する場合の処理の流れを示す図である。

【図２４】

本発明の第７の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、サイズ毎もしくは個数毎にログを分割保存する場合の処理の流れを示す図である。

【図２５】

本発明の第８の実施形態にかかるログ取得方法を実現するソフトウェア評価システムにおいて、取得したログを一定数だけメモリに保存する場合のメモリの概要図である。

【図２６】

本発明の第８の実施形態にかかるログ取得方法を実現するソフトウェア評価システムにおいて、取得したログを一定数だけメモリに保存する場合のログ取得の流れを示す図である。

【符号の説明】

１：ＣＰＵ

２：チップセット

３：ＲＡＭ

４：ハードディスクコントローラ

５：ディスプレイコントローラ

６：ハードディスクドライブ

７：ＣＤ－ＲＯＭドライブ

８：ディスプレイ

１１：ＣＰＵとチップセットを繋ぐ信号線

１２：チップセットとＲＡＭを繋ぐ信号線

１３：チップセットと各種周辺機器とを繋ぐ周辺機器バス

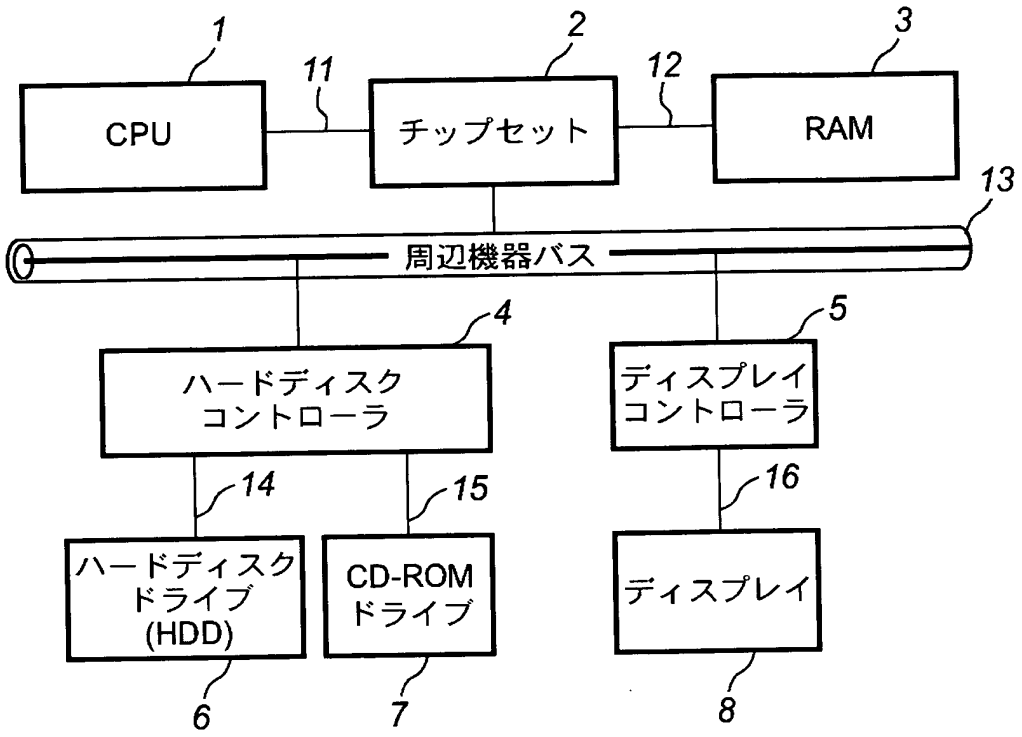
１４：ハードディスクコントローラとハードディスクドライブを繋ぐ信号線

１５：ハードディスクコントローラとＣＤ－ＲＯＭドライブを繋ぐ信号線

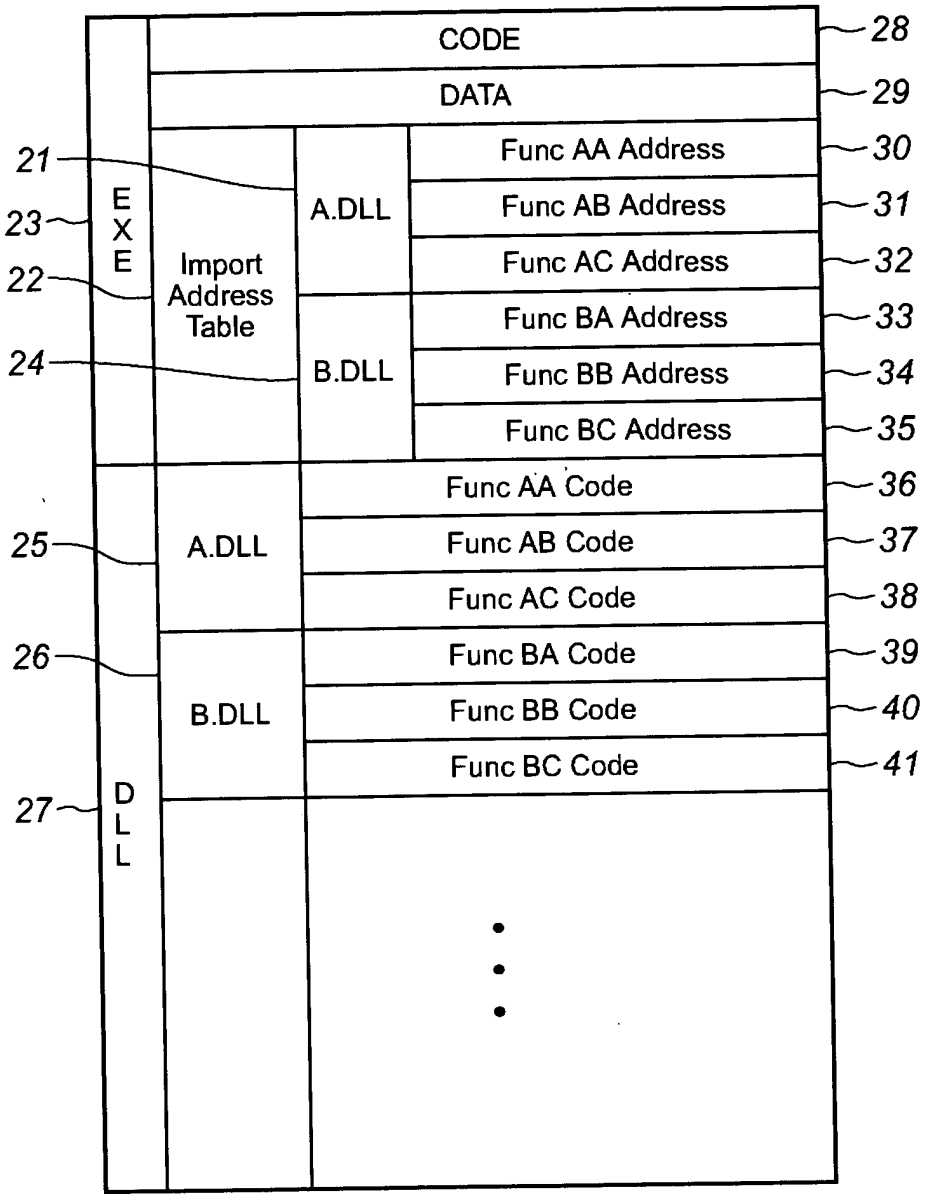
１６：ディスプレイコントローラとディスプレイを繋ぐ信号線

【書類名】　図面

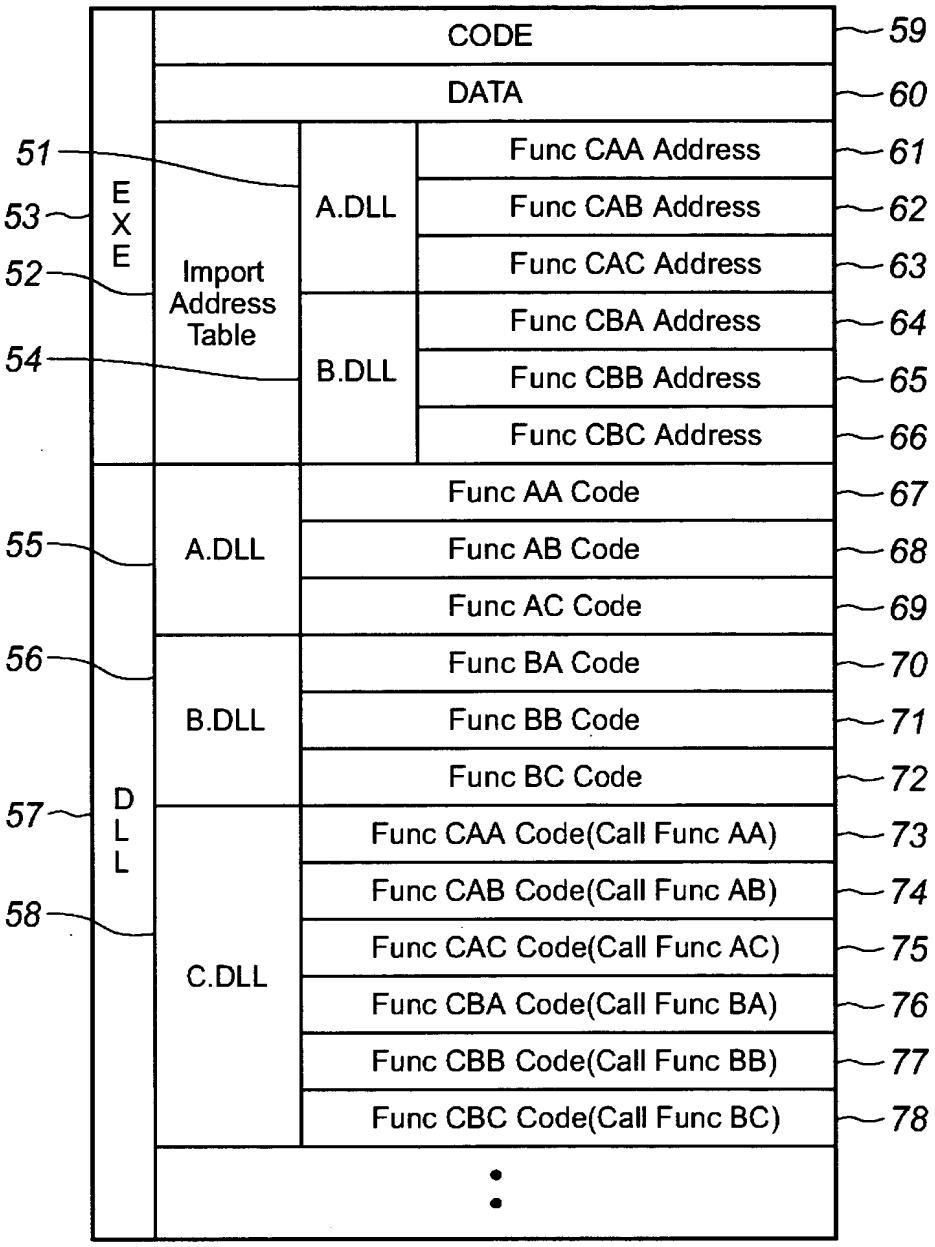
【図１】


1
2
3
11
12
CPU
チップセット
RAM
13
周辺機器バス
4
5
ハードディスク
コントローラ
ディスプレイ
コントローラ
14
15
16
ハードディスク
ドライブ
(HDD)
CD-ROM
ドライブ
ディスプレイ
6
7
8

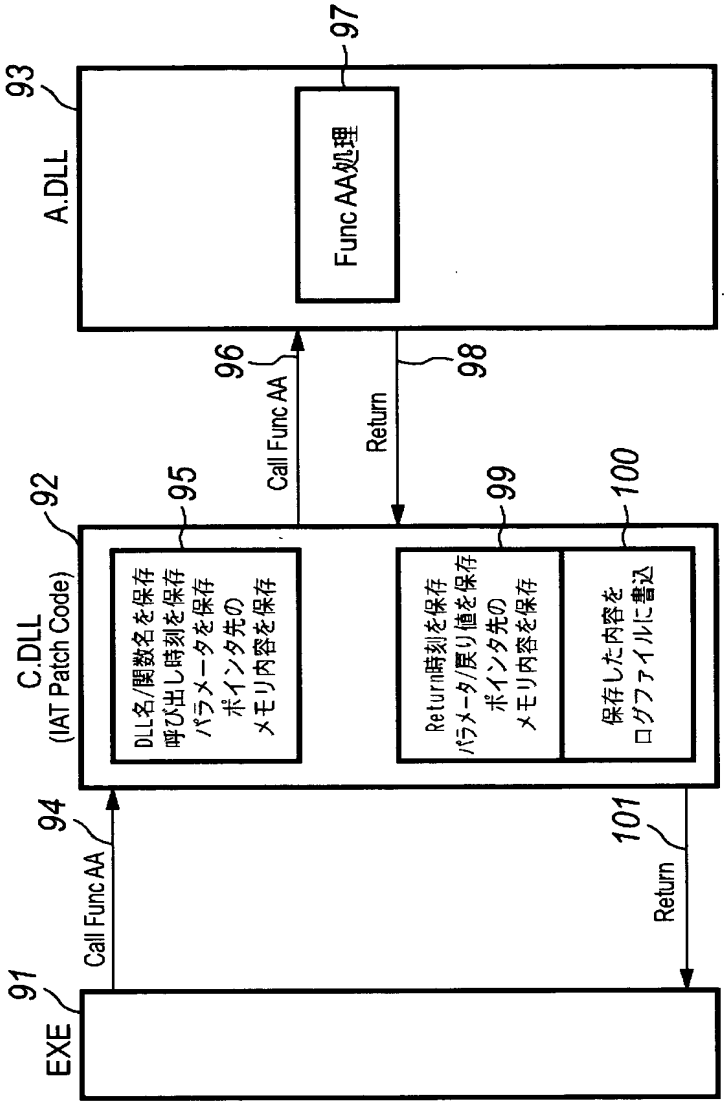
【図２】


CODE
DATA
EXE
Import Address Table
A.DLL
B.DLL
Func AA Address
Func AB Address
Func AC Address
Func BA Address
Func BB Address
Func BC Address
DLL
A.DLL
B.DLL
Func AA Code
Func AB Code
Func AC Code
Func BA Code
Func BB Code
Func BC Code
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

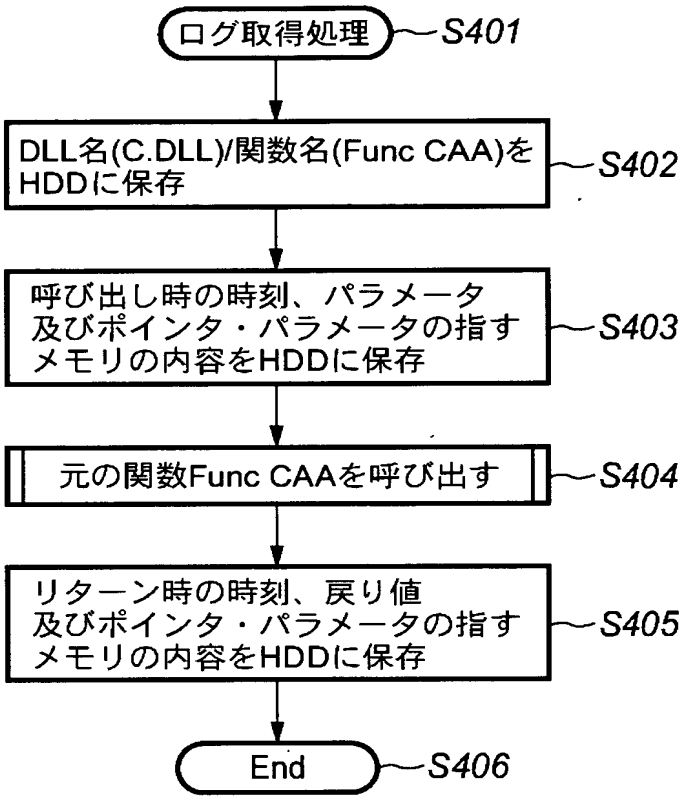
【図３】


CODE
DATA
Import Address Table
A.DLL
Func CAA Address
Func CAB Address
Func CAC Address
B.DLL
Func CBA Address
Func CBB Address
Func CBC Address
EXE
A.DLL
Func AA Code
Func AB Code
Func AC Code
B.DLL
Func BA Code
Func BB Code
Func BC Code
C.DLL
Func CAA Code(Call Func AA)
Func CAB Code(Call Func AB)
Func CAC Code(Call Func AC)
Func CBA Code(Call Func BA)
Func CBB Code(Call Func BB)
Func CBC Code(Call Func BC)
DLL
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78

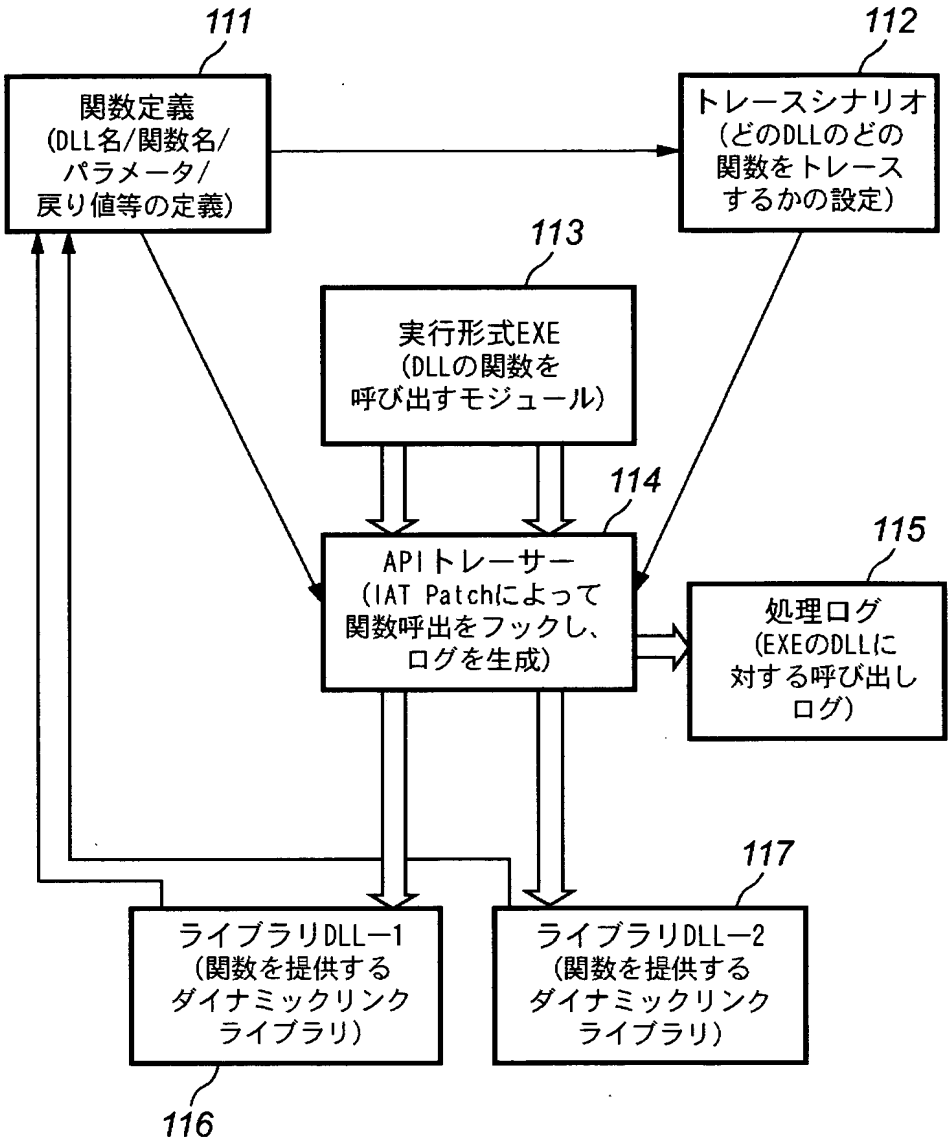
【図４Ａ】


97
93
A.DLL
Func AA処理
96
Call Func AA
Return
98
92
95
99
100
C.DLL
(IAT Patch Code)
DLL名/関数名を保存
呼び出し時刻を保存
パラメータを保存
ポインタ先の
メモリ内容を保存
Return時刻を保存
パラメータ/戻り値を保存
ポインタ先の
メモリ内容を保存
保存した内容を
ログファイルに書込
94
Call Func AA
101
Return
91
EXE

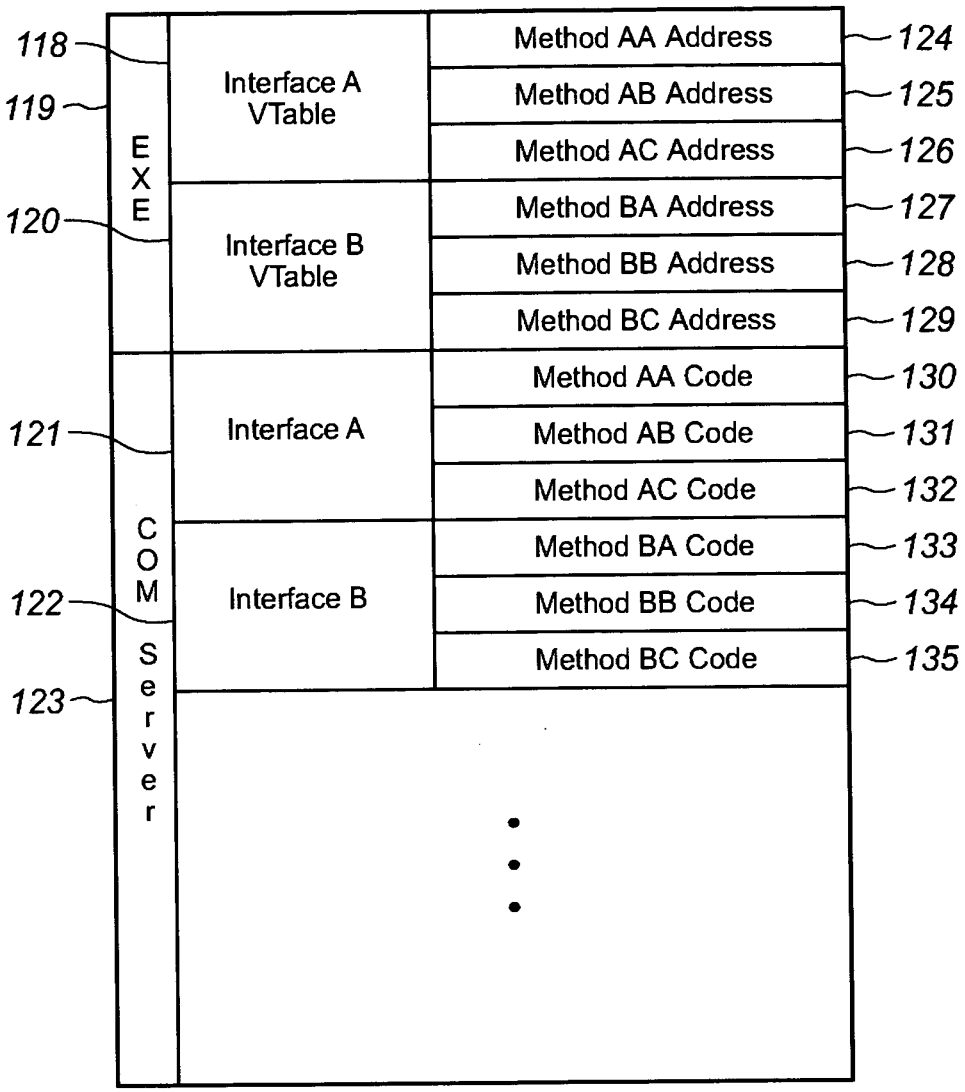
【図４Ｂ】


ログ取得処理
S401
DLL名(C.DLL)/関数名(Func CAA)を
HDDに保存
S402
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S403
元の関数Func CAAを呼び出す
S404
リターン時の時刻、戻り値
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S405
End
S406

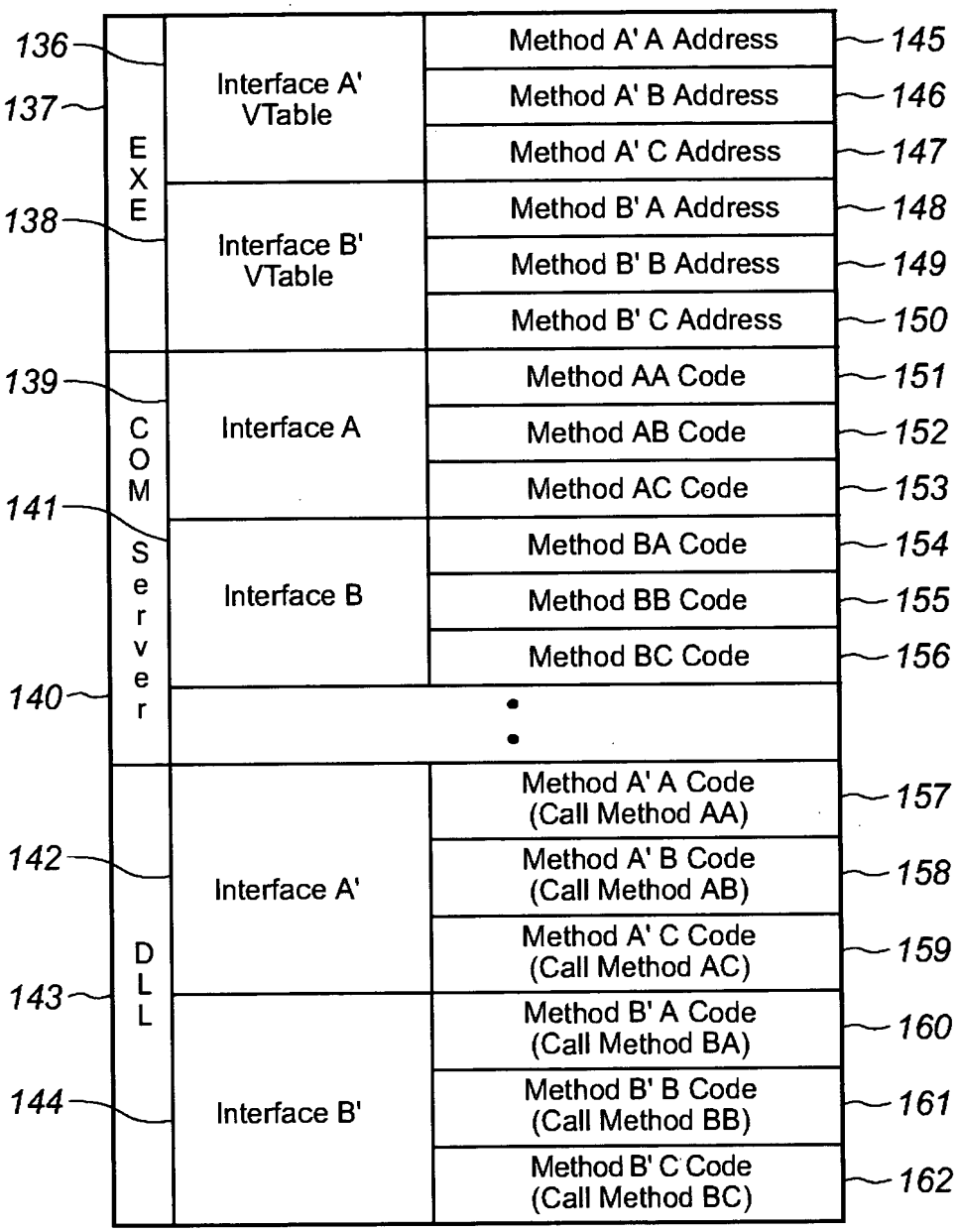
【図５】


111
関数定義
(DLL名/関数名/
パラメータ/
戻り値等の定義)
112
トレースシナリオ
(どのDLLのどの
関数をトレース
するかの設定)
113
実行形式EXE
(DLLの関数を
呼び出すモジュール)
114
115
APIトレーサー
(IAT Patchによって
関数呼出をフックし、
ログを生成)
処理ログ
(EXEのDLLに
対する呼び出し
ログ)
117
ライブラリDLL−1
(関数を提供する
ダイナミックリンク
ライブラリ)
ライブラリDLL−2
(関数を提供する
ダイナミックリンク
ライブラリ)
116

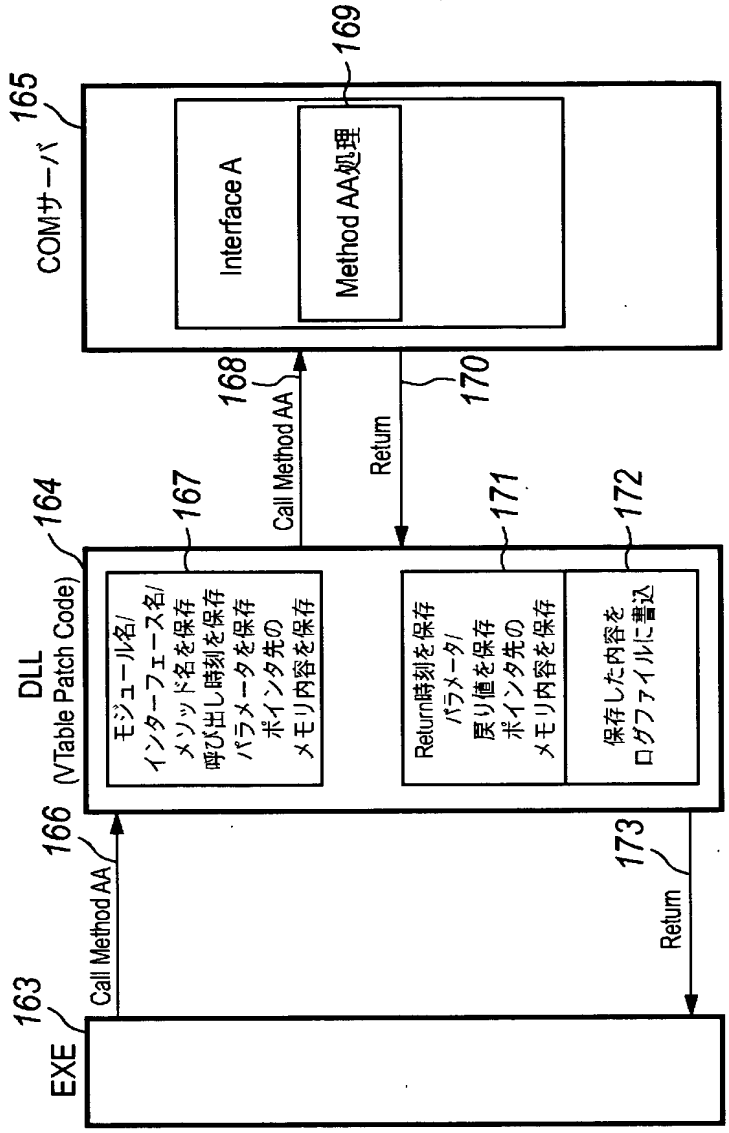
【図６】


118
119
120
121
122
123
EXE
COM Server
Interface A VTable
Interface B VTable
Interface A
Interface B
Method AA Address
Method AB Address
Method AC Address
Method BA Address
Method BB Address
Method BC Address
Method AA Code
Method AB Code
Method AC Code
Method BA Code
Method BB Code
Method BC Code
124
125
126
127
128
129
130
131
132
133
134
135

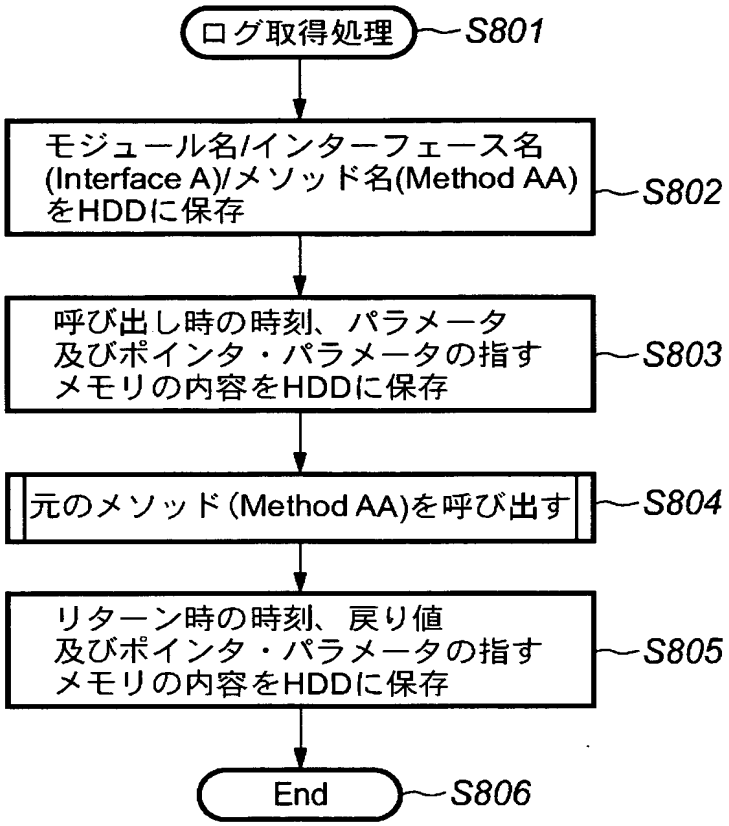
【図７】


136
137
138
139
141
140
142
143
144
EXE
COM Server
DLL
Interface A' VTable
Interface B' VTable
Interface A
Interface B
Interface A'
Interface B'
Method A' A Address
Method A' B Address
Method A' C Address
Method B' A Address
Method B' B Address
Method B' C Address
Method AA Code
Method AB Code
Method AC Code
Method BA Code
Method BB Code
Method BC Code
Method A' A Code (Call Method AA)
Method A' B Code (Call Method AB)
Method A' C Code (Call Method AC)
Method B' A Code (Call Method BA)
Method B' B Code (Call Method BB)
Method B' C Code (Call Method BC)
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162

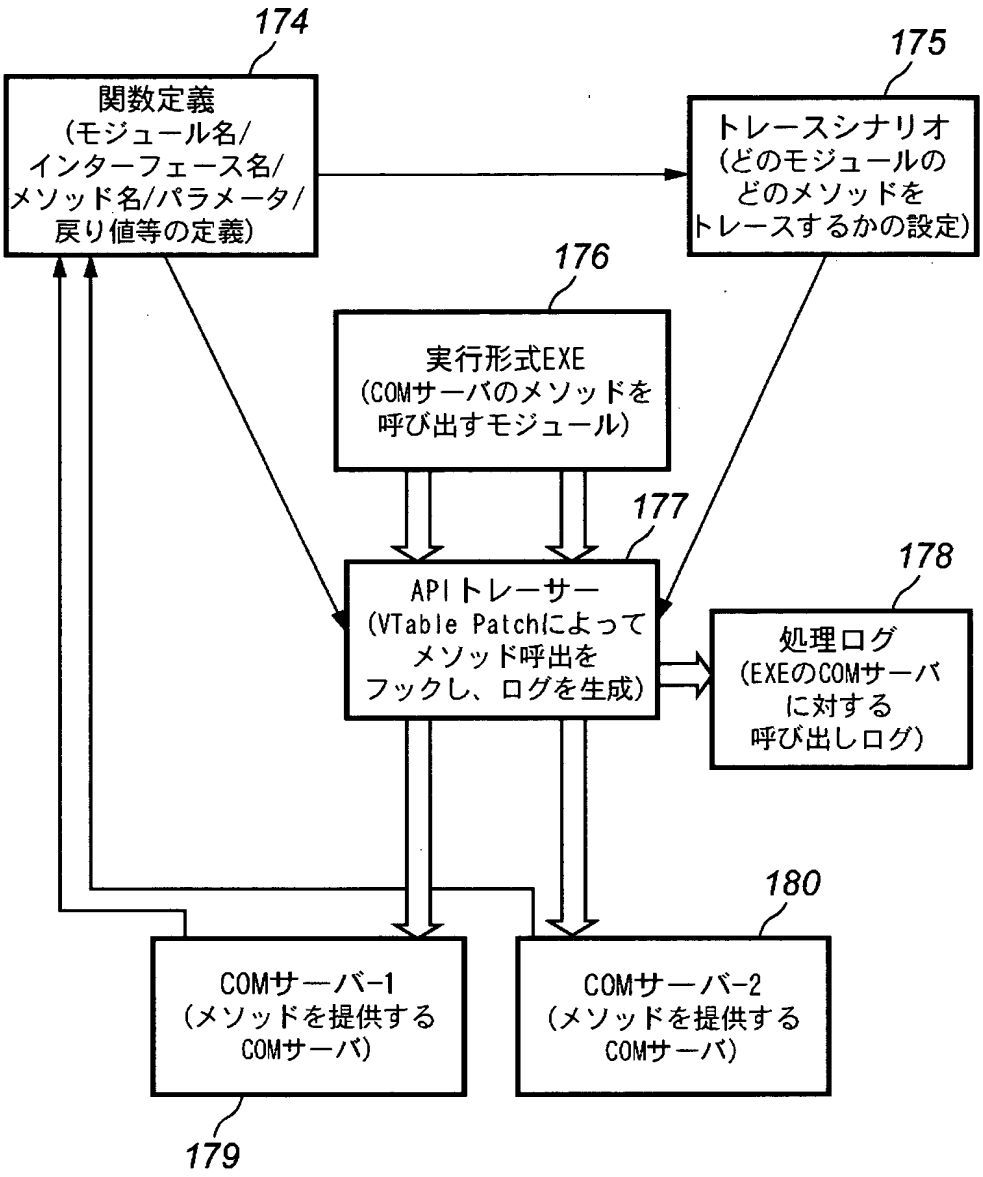
【図８Ａ】


165
COMサーバ
169
Interface A
Method AA処理
168
Call Method AA
170
Return
164
DLL
(VTable Patch Code)
167
モジュール名/
インターフェース名/
メソッド名を保存
呼び出し時刻を保存
パラメータを保存
ポインタ先の
メモリ内容を保存
171
Return時刻を保存
パラメータ/
戻り値を保存
ポインタ先の
メモリ内容を保存
172
保存した内容を
ログファイルに書込
166
Call Method AA
173
Return
163
EXE

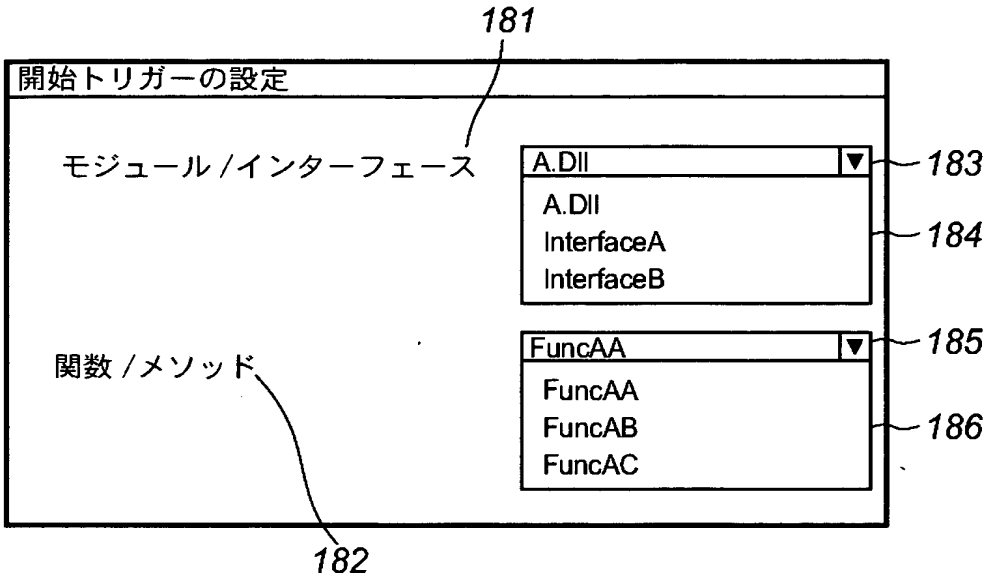
【図８Ｂ】


ログ取得処理
S801
モジュール名/インターフェース名(Interface A)/メソッド名(Method AA)をHDDに保存
S802
呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をHDDに保存
S803
元のメソッド(Method AA)を呼び出す
S804
リターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をHDDに保存
S805
End
S806

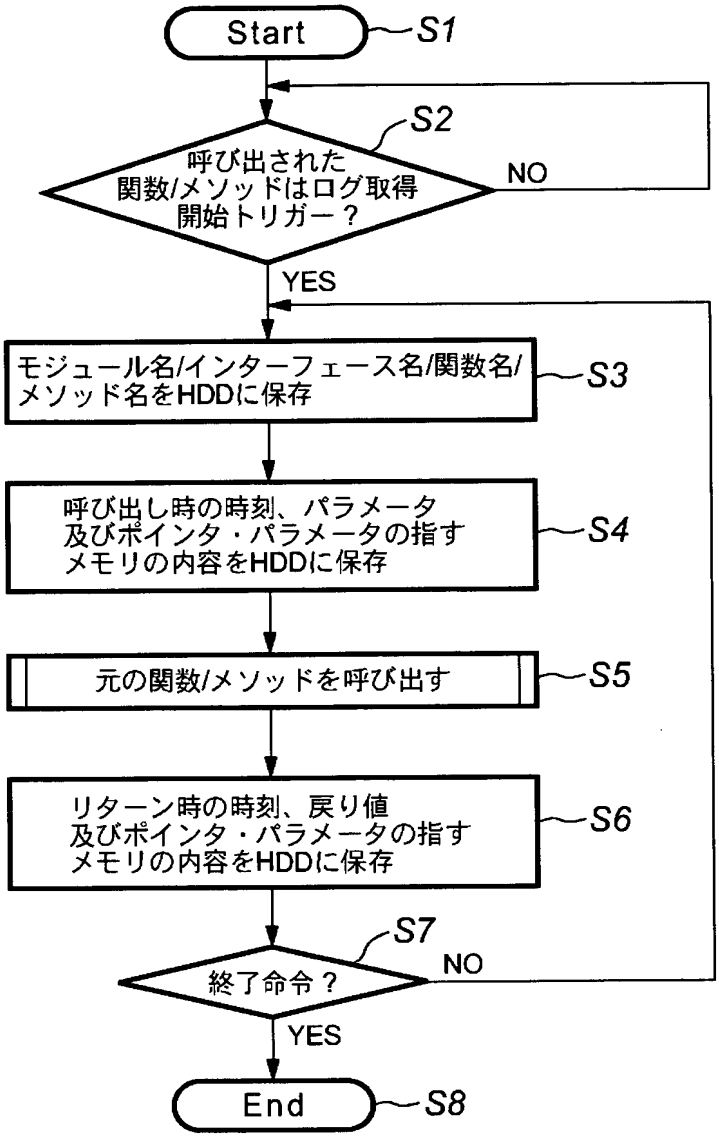
【図９】


174
関数定義
(モジュール名/
インターフェース名/
メソッド名/パラメータ/
戻り値等の定義)
175
トレースシナリオ
(どのモジュールの
どのメソッドを
トレースするかの設定)
176
実行形式EXE
(COMサーバのメソッドを
呼び出すモジュール)
177
APIトレーサー
(VTable Patchによって
メソッド呼出を
フックし、ログを生成)
178
処理ログ
(EXEのCOMサーバ
に対する
呼び出しログ)
180
COMサーバ-1
(メソッドを提供する
COMサーバ)
COMサーバ-2
(メソッドを提供する
COMサーバ)
179

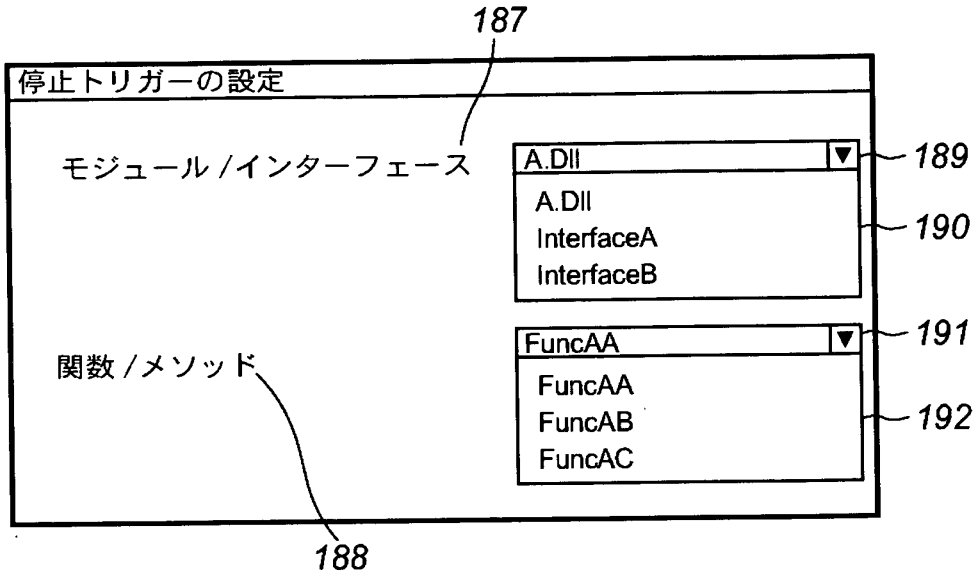
【図１０】


181
開始トリガーの設定
モジュール /インターフェース
A.Dll
183
A.Dll
InterfaceA
InterfaceB
184
FuncAA
185
関数 /メソッド
FuncAA
FuncAB
FuncAC
186
182

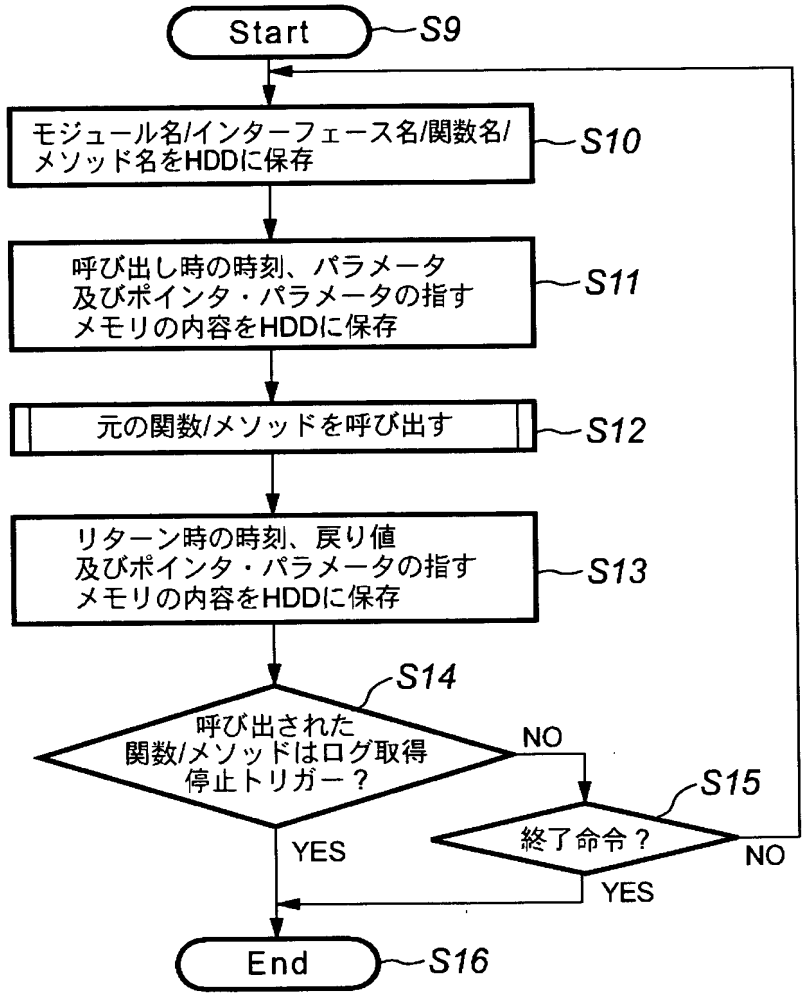
【図１１】


Start
S1
S2
呼び出された
関数/メソッドはログ取得
開始トリガー？
NO
YES
モジュール名/インターフェース名/関数名/
メソッド名をHDDに保存
S3
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S4
元の関数/メソッドを呼び出す
S5
リターン時の時刻、戻り値
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S6
S7
終了命令？
NO
YES
End
S8

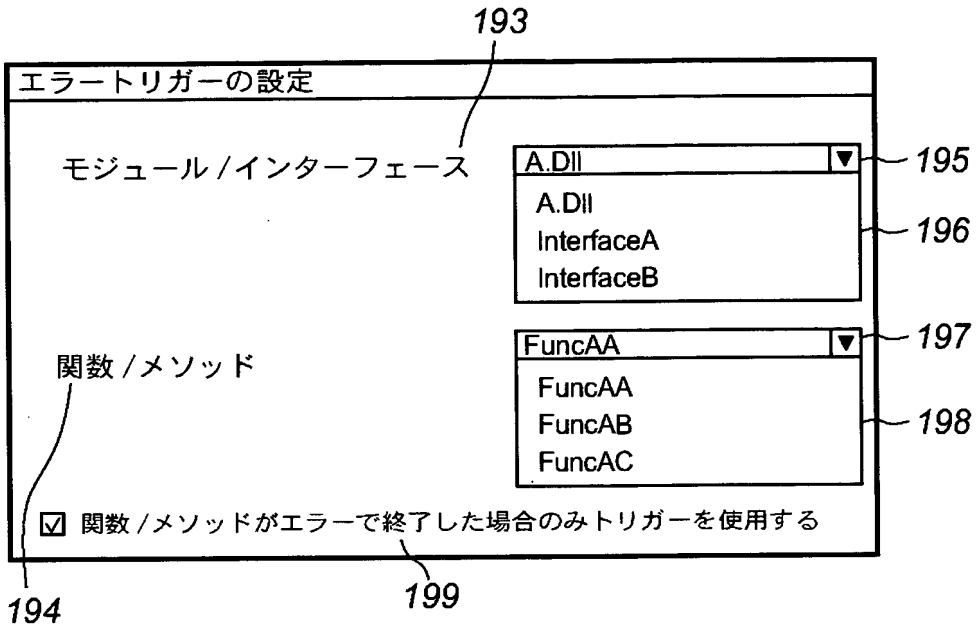
【図１２】


187
停止トリガーの設定
モジュール /インターフェース
A.Dll
189
A.Dll
InterfaceA
InterfaceB
190
FuncAA
191
関数 /メソッド
FuncAA
FuncAB
FuncAC
192
188

【図１３】


Start
S9
モジュール名/インターフェース名/関数名/
メソッド名をHDDに保存
S10
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S11
元の関数/メソッドを呼び出す
S12
リターン時の時刻、戻り値
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S13
S14
呼び出された
関数/メソッドはログ取得
停止トリガー？
NO
YES
S15
終了命令？
NO
YES
End
S16

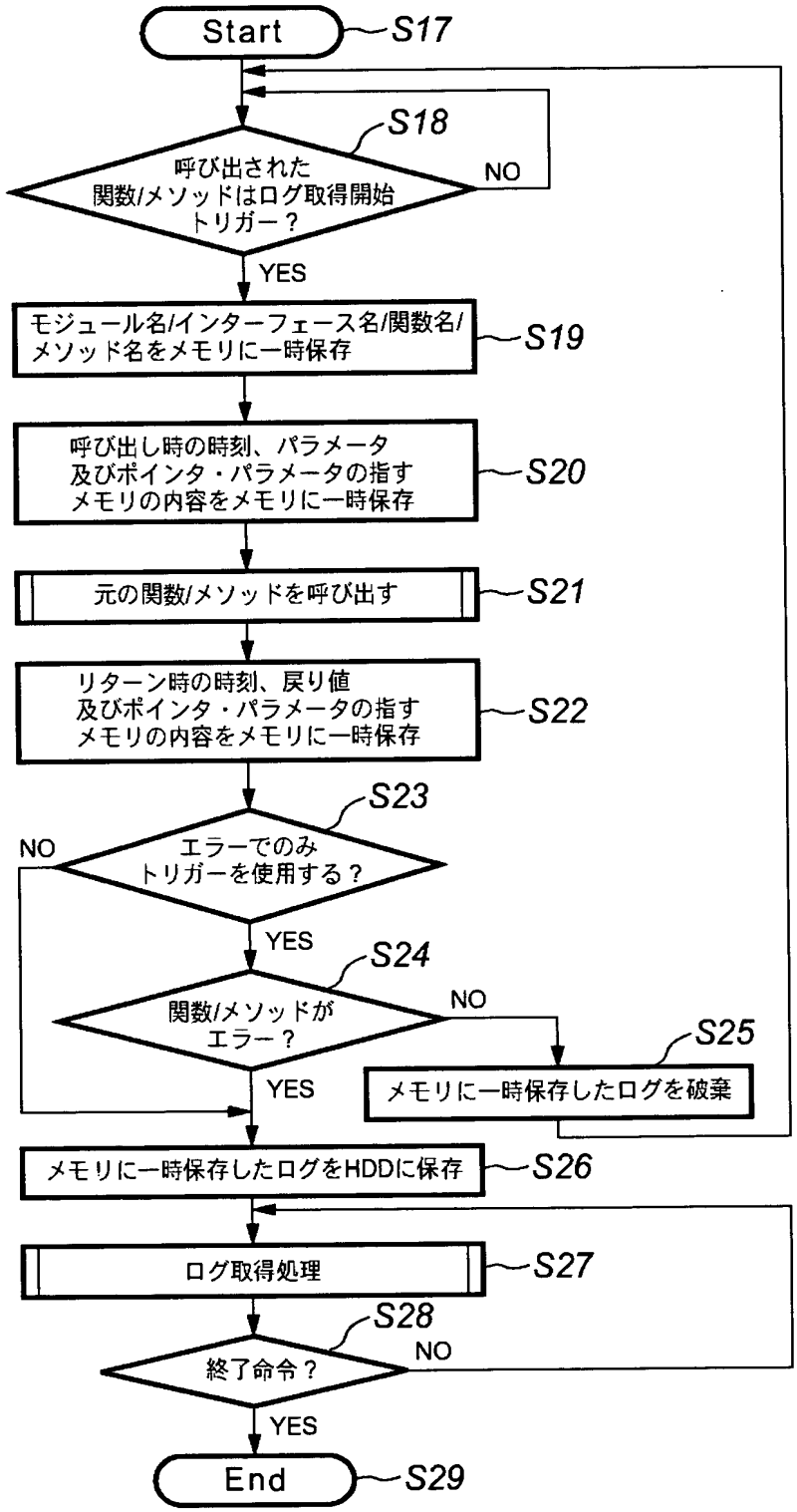
【図１４】


193
エラートリガーの設定
モジュール/インターフェース
A.Dll
A.Dll
InterfaceA
InterfaceB
195
196
関数/メソッド
FuncAA
FuncAA
FuncAB
FuncAC
197
198
関数/メソッドがエラーで終了した場合のみトリガーを使用する
194
199

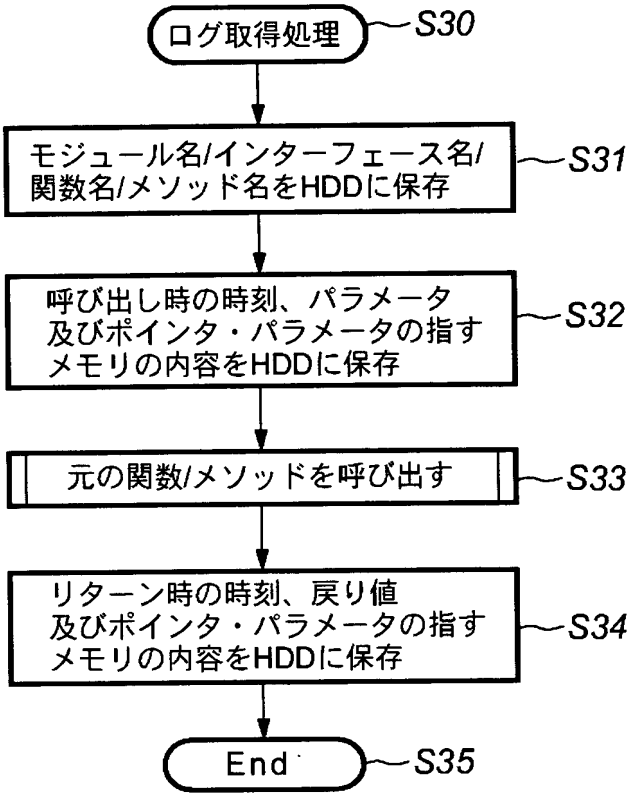
【図１５】

モジュール名：	A. DLL
関数名：	FuncAA
引数：	DWORD dwID： Err>100
戻り値：	DWORD dwRet： Err==0
モジュール名：	A. DLL
関数名：	FuncAB
引数：	DWORD dwHandle： Err==0
戻り値：	int nRet： Err<=-1
モジュール名：	B. DLL
インターフェース名：	InterfaceA
メソッド名：	MethodAA
引数：	DWORD dwID： Err>100
戻り値：	DWORD dwHandle： Err==0
モジュール名：	B. DLL
インターフェース名：	InterfaceA
メソッド名：	MethodAB
引数：	DWORD dwID： Err<=0
戻り値：	DWORD dwRet： Err !=0
モジュール名：	B. DLL
インターフェース名：	InterfaceB
メソッド名：	MethodBA
引数：	DWORD dwID： Err>=0
戻り値：	DWORD dwRet： Err !=0

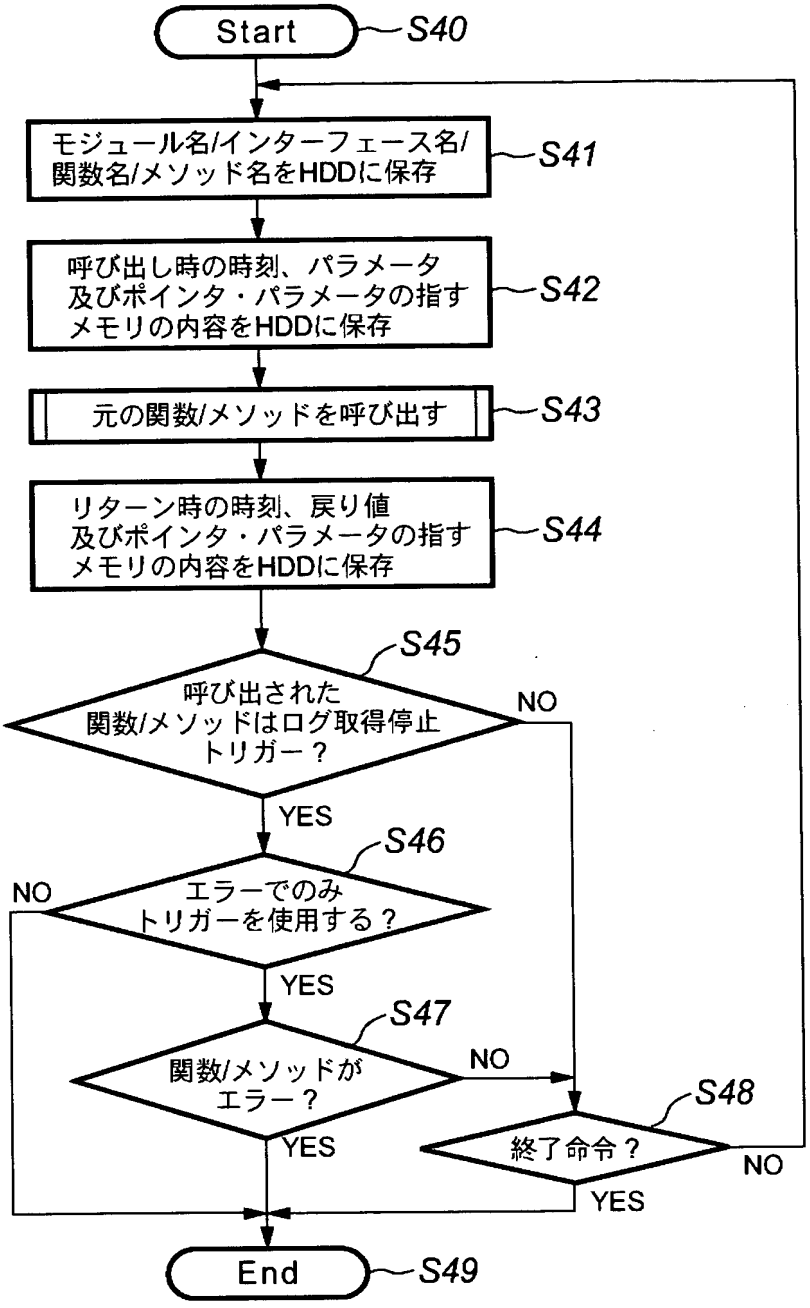
【図１６】


Start
S17
S18
呼び出された
関数/メソッドはログ取得開始
トリガー？
NO
YES
モジュール名/インターフェース名/関数名/
メソッド名をメモリに一時保存
S19
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指す
メモリの内容をメモリに一時保存
S20
元の関数/メソッドを呼び出す
S21
リターン時の時刻、戻り値
及びポインタ・パラメータの指す
メモリの内容をメモリに一時保存
S22
S23
エラーでのみ
トリガーを使用する？
NO
YES
S24
関数/メソッドが
エラー？
NO
S25
メモリに一時保存したログを破棄
YES
メモリに一時保存したログをHDDに保存
S26
ログ取得処理
S27
S28
終了命令？
NO
YES
End
S29

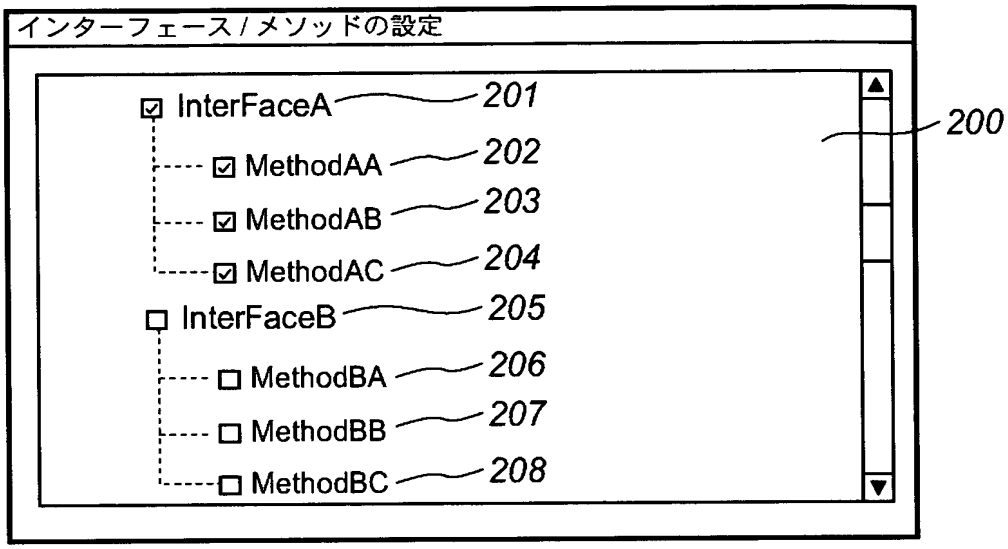
【図１７】


ログ取得処理
S30
モジュール名/インターフェース名/
関数名/メソッド名をHDDに保存
S31
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S32
元の関数/メソッドを呼び出す
S33
リターン時の時刻、戻り値
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S34
End
S35

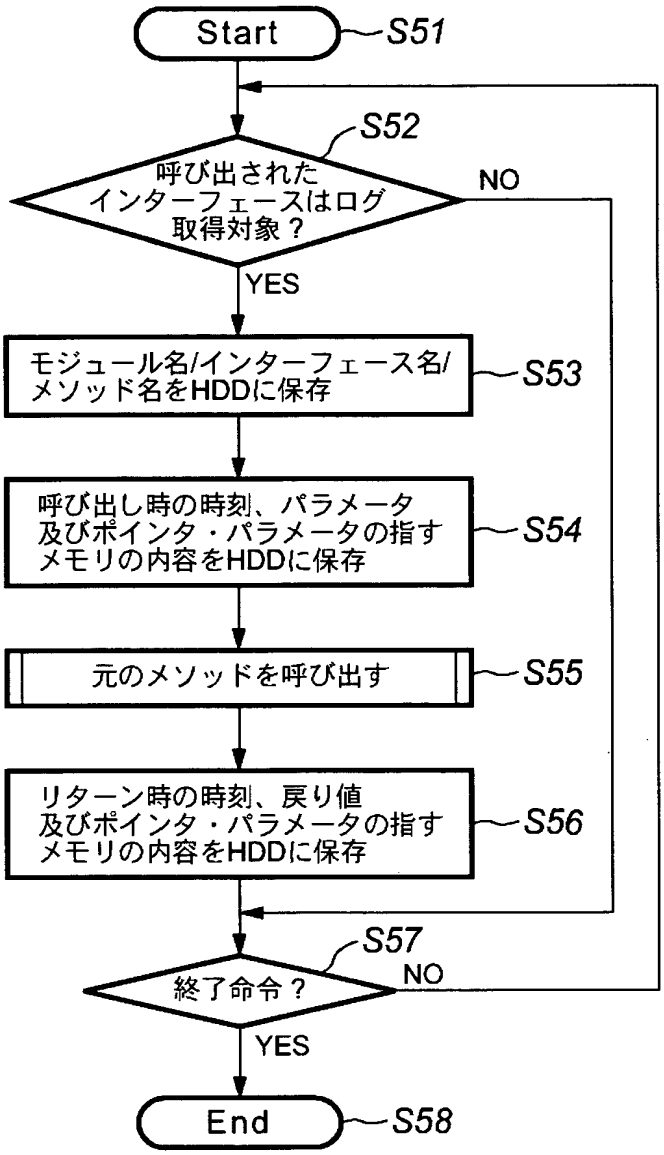
【図１８】


Start
S40
モジュール名/インターフェース名/
関数名/メソッド名をHDDに保存
S41
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S42
元の関数/メソッドを呼び出す
S43
リターン時の時刻、戻り値
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S44
S45
呼び出された
関数/メソッドはログ取得停止
トリガー？
NO
YES
S46
NO
エラーでのみ
トリガーを使用する？
YES
S47
関数/メソッドが
エラー？
NO
YES
S48
終了命令？
NO
YES
End
S49

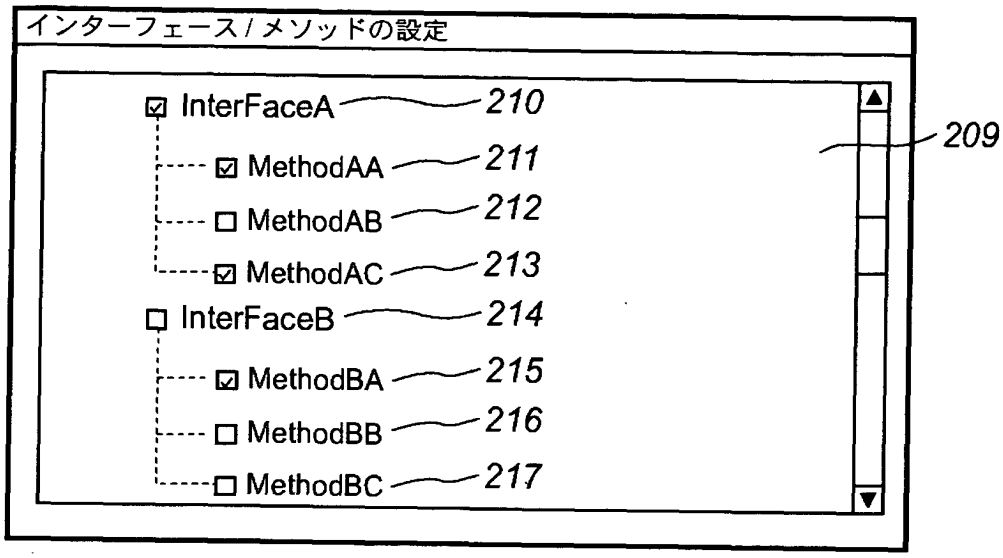
【図１９】


インターフェース / メソッドの設定
InterFaceA
201
MethodAA
202
MethodAB
203
MethodAC
204
InterFaceB
205
MethodBA
206
MethodBB
207
MethodBC
208
200

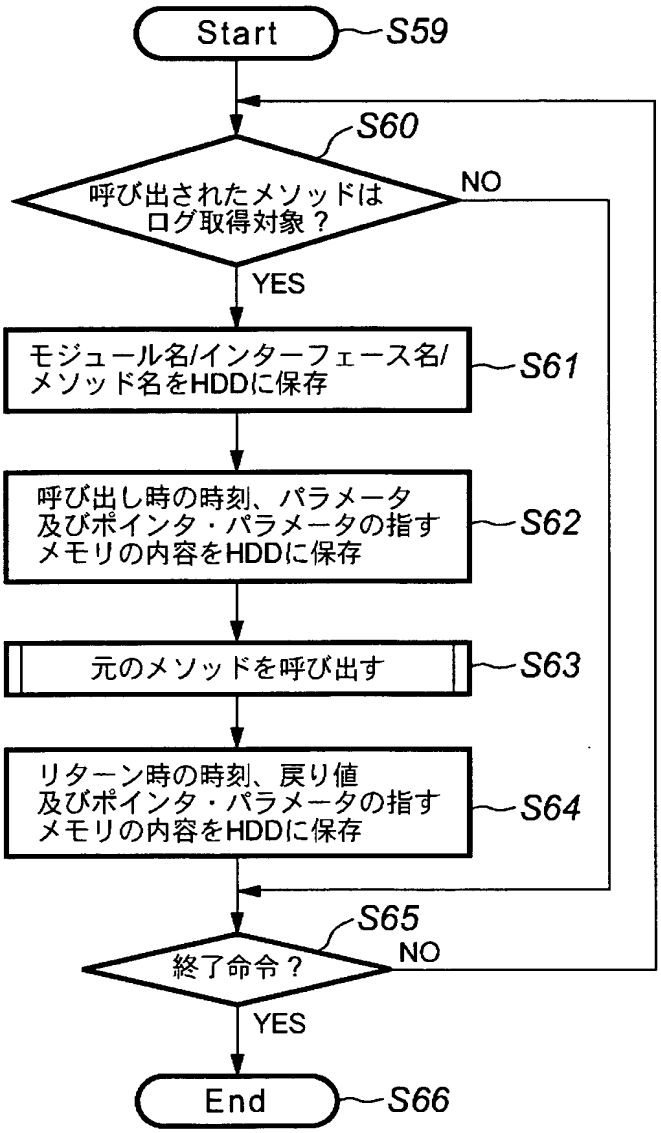
【図２０】


Start
S51
S52
呼び出された
インターフェースはログ
取得対象？
NO
YES
モジュール名/インターフェース名/
メソッド名をHDDに保存
S53
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S54
元のメソッドを呼び出す
S55
リターン時の時刻、戻り値
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S56
S57
終了命令？
NO
YES
End
S58

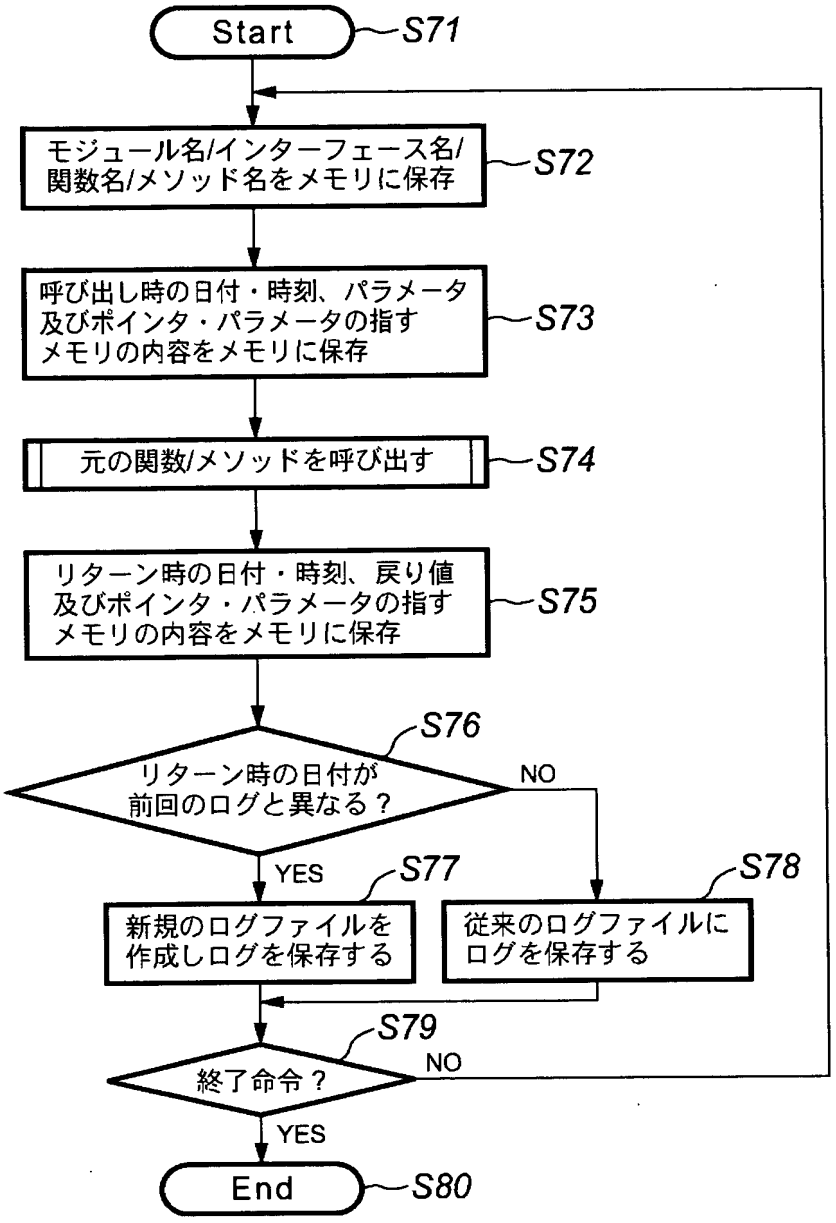
【図２１】


インターフェース/メソッドの設定
InterFaceA
210
MethodAA
211
MethodAB
212
MethodAC
213
InterFaceB
214
MethodBA
215
MethodBB
216
MethodBC
217
209

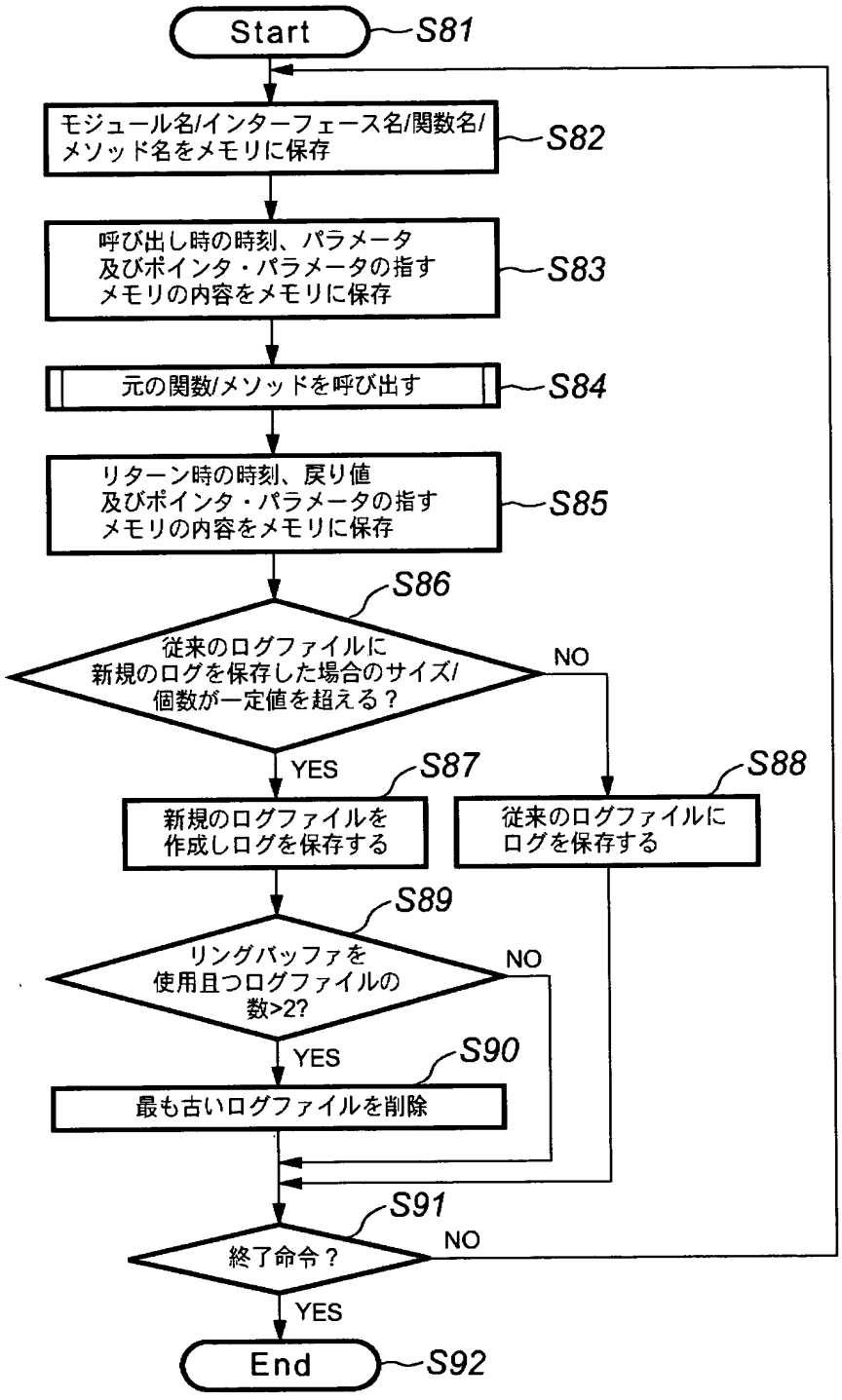
【図２２】


Start
S59
S60
呼び出されたメソッドは
ログ取得対象？
NO
YES
モジュール名/インターフェース名/
メソッド名をHDDに保存
S61
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S62
元のメソッドを呼び出す
S63
リターン時の時刻、戻り値
及びポインタ・パラメータの指す
メモリの内容をHDDに保存
S64
S65
終了命令？
NO
YES
End
S66

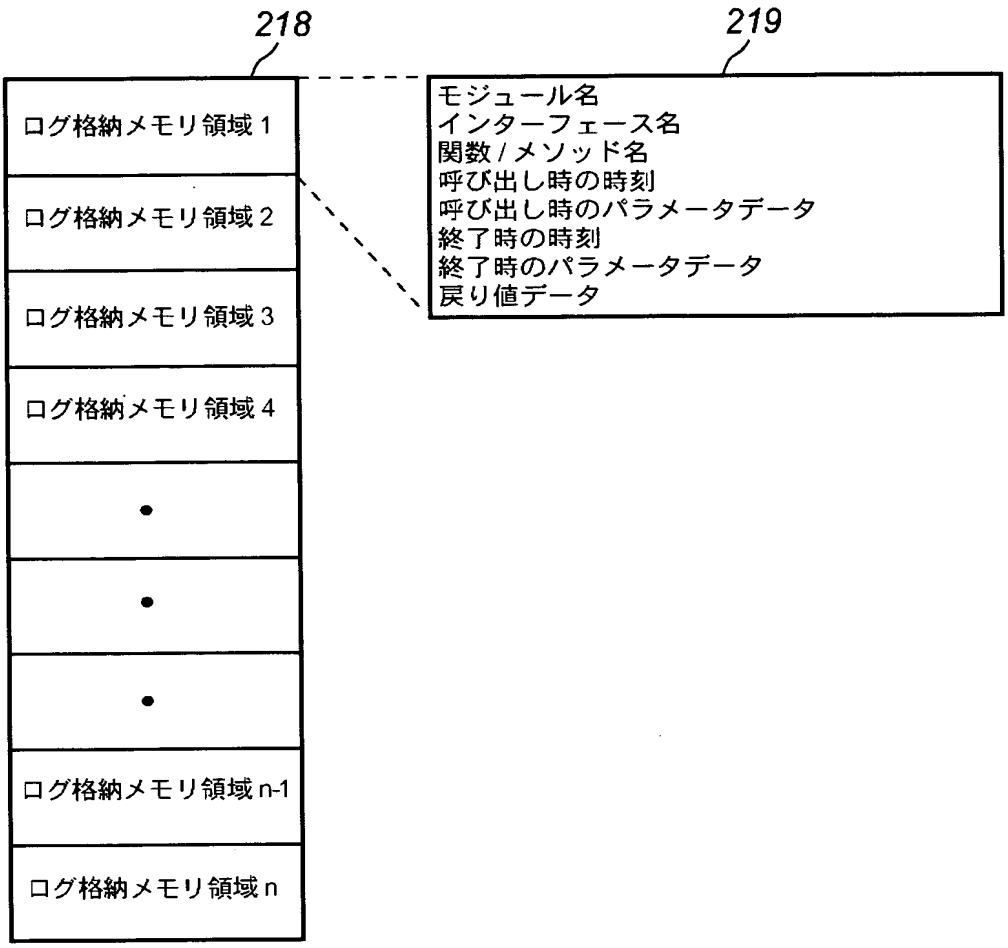
【図２３】


Start
S71
モジュール名/インターフェース名/関数名/メソッド名をメモリに保存
S72
呼び出し時の日付・時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をメモリに保存
S73
元の関数/メソッドを呼び出す
S74
リターン時の日付・時刻、戻り値及びポインタ・パラメータの指すメモリの内容をメモリに保存
S75
S76
リターン時の日付が前回のログと異なる？
NO
YES
S77
新規のログファイルを作成しログを保存する
S78
従来のログファイルにログを保存する
S79
終了命令？
NO
YES
End
S80

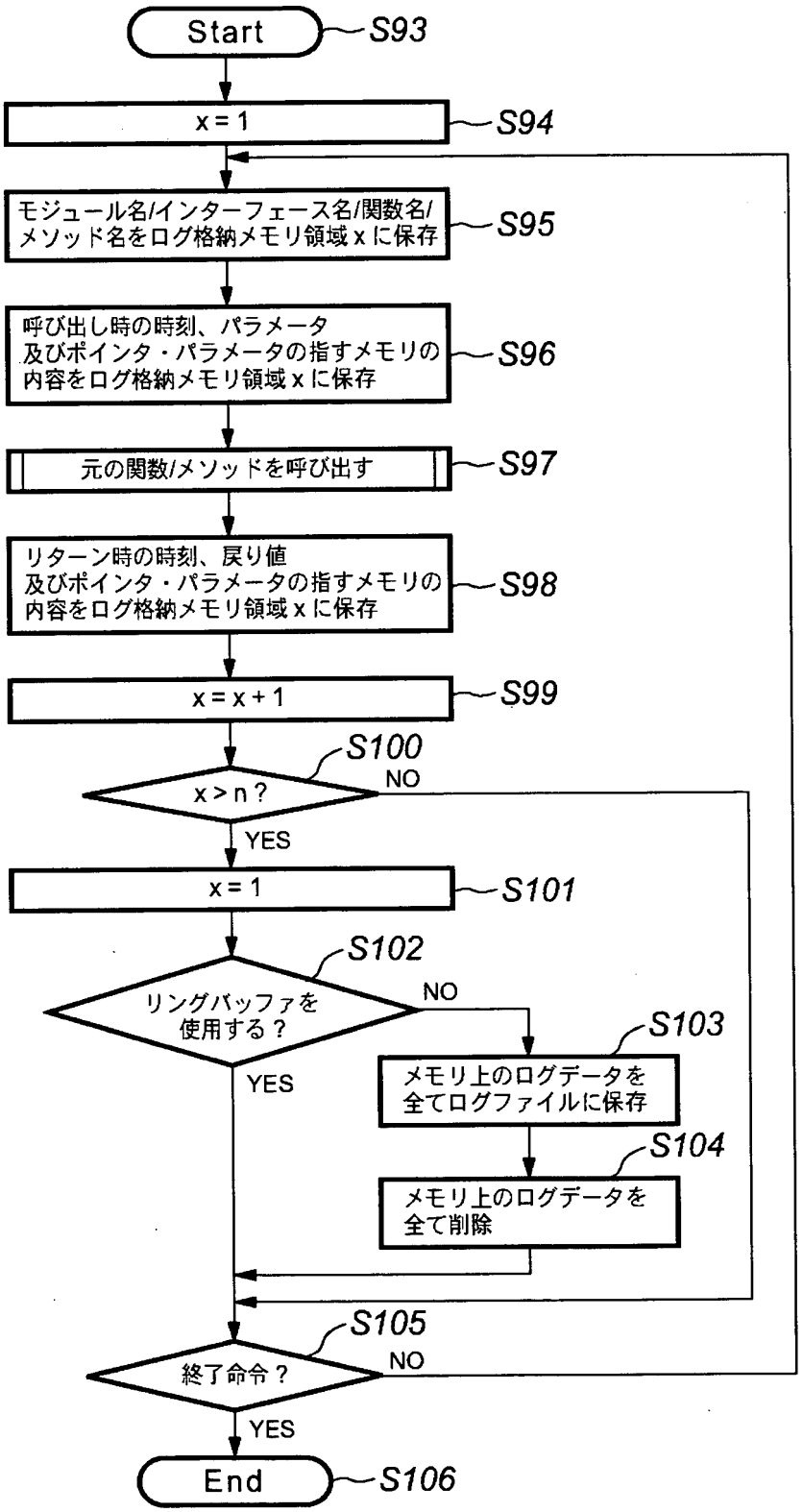
【図２４】


Start
S81
モジュール名/インターフェース名/関数名/
メソッド名をメモリに保存
S82
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指す
メモリの内容をメモリに保存
S83
元の関数/メソッドを呼び出す
S84
リターン時の時刻、戻り値
及びポインタ・パラメータの指す
メモリの内容をメモリに保存
S85
S86
従来のログファイルに
新規のログを保存した場合のサイズ/
個数が一定値を超える？
NO
YES
S87
新規のログファイルを
作成しログを保存する
S88
従来のログファイルに
ログを保存する
S89
リングバッファを
使用且つログファイルの
数>2?
NO
YES
S90
最も古いログファイルを削除
S91
終了命令？
NO
YES
End
S92

【図２５】


218
219
ログ格納メモリ領域１
ログ格納メモリ領域２
ログ格納メモリ領域３
ログ格納メモリ領域４
ログ格納メモリ領域ｎ-1
ログ格納メモリ領域ｎ
モジュール名
インターフェース名
関数／メソッド名
呼び出し時の時刻
呼び出し時のパラメータデータ
終了時の時刻
終了時のパラメータデータ
戻り値データ

【図２６】


Start
S93
x = 1
S94
モジュール名/インターフェース名/関数名/
メソッド名をログ格納メモリ領域 x に保存
S95
呼び出し時の時刻、パラメータ
及びポインタ・パラメータの指すメモリの
内容をログ格納メモリ領域 x に保存
S96
元の関数/メソッドを呼び出す
S97
リターン時の時刻、戻り値
及びポインタ・パラメータの指すメモリの
内容をログ格納メモリ領域 x に保存
S98
x = x + 1
S99
S100
x > n ?
NO
YES
x = 1
S101
S102
リングバッファを
使用する？
NO
YES
S103
メモリ上のログデータを
全てログファイルに保存
S104
メモリ上のログデータを
全て削除
S105
終了命令？
NO
YES
End
S106

【書類名】　要約書

【要約】

【課題】　ソフトウェアの処理ログを容易に取得でき、かつ、バグの解析のための工数を削減することが可能なログ取得方法を提供する。

【解決手段】　関数（FuncAA）を備えるプログラム９１において、実行中のログを取得するログ取得方法であって、ロードされた関数（FuncAA）のアドレスを、ログ取得のための関数（９２）のアドレスに書き換える工程と、関数（FuncAA）を選択する工程と、を備え、ログ取得のための関数（９２）は、関数（FuncAA）のアドレスを呼び出し（９６）、該所定の処理を実行させ（９７）、受け取った実行結果（９８）をプログラム９１に受け渡す工程（１０１）と、前記選択された関数（FuncAA）のアドレスを呼び出す際の所定の情報を記録する工程（９５、１００））と、前記選択された関数（FuncAA）の前記実行結果を受け取った際の所定の情報を記録する工程（９９、１００）とを備える。

【選択図】　図４Ａ

特願２００２－１９１１２９

出　願　人　履　歴　情　報

識別番号　　［０００００１００７］

１．変更年月日　　１９９０年　８月３０日
［変更理由］　　新規登録
住　所　　東京都大田区下丸子３丁目３０番２号
氏　名　　キヤノン株式会社